1992

Nonlinear Inelastic Finite Element Analysis of Reinforced Concrete Structures With Emphasis on Shear and Torsion. (Volumes I and II).

Ananth Ramaswamy
Louisiana State University and Agricultural & Mechanical College

Order Number 9301099

Nonlinear inelastic finite element analysis of reinforced concrete structures with emphasis on shear and torsion. (Volumes I and II)

Ramaswamy, Ananth, Ph.D.

The Louisiana State University and Agricultural and Mechanical Col., 1992

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

NONLINEAR INELASTIC FINITE ELEMENT ANALYSIS OF REINFORCED CONCRETE STRUCTURES WITH EMPHASIS ON SHEAR AND TORSION
Volume I

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Civil Engineering

by
Ananth Ramaswamy
B.Tech., Indian Institute of Technology, 1985
M.S. University of California, 1986
May, 1992

Acknowledgements

This study was conducted under the supervision of Dr.F. Barzegar, formerly Assistant Professor of Civil Engineering, LSU. The work was completed under the supervision of Dr.G.Z. Voyiadjis, Professor of Civil Engineering, LSU. I wish to thank both of them for their guidance, encouragement, and the many useful discussions I had with them while conducting this study.

I would also like to thank the members of my doctoral advisory committee- Professor R.R. Avent, Professor V.K. Gopu, Professor S.S. Iyengar, Professor P.N. Kirk, and Dr.P. Zuraski for reviewing this dissertation and offering their helpful suggestions.

Financial support provided by the Department of Civil Engineering, LSU and the National Science Foundation while conducting this study is most gratefully acknowledged. A special word of thanks to the Faculty and staff, Department of Civil Engineering, LSU, for providing computational hardware periferals while conducting this study.

This study was conducted using the IBM 3090/MVS mainframe computer. The funds provided by the university for this purpose is gratefully acknowledged. A special word of thanks to Prof. W.F. Beyer, Director of SNCC, LSU, for his cooperation and encouragement. The help provided by the SNCC staff, in particular, Dr.M. Foroozesh, Mr.F.J. Quinn, and Ms.D.D. Sutton is also gratefully acknowledged.

I wish to thank Dr.C. Channakeshava, Research Associate at LSU, for the many useful discussions I have had with him, during the course of this study. I also wish to thank my colleagues- Mr.S. Maddipudi, Mr.E. Rainer, Mr.A. Venson, Mr.A. Taher, Mr P. Kurup, Dr.H. Pentas, Mr.A. Puppala and Dr.K. Rebello; who have made my graduate school experience at LSU a pleasant one. Mr.A. Venson's knowledge and help in using 'LATEX' deserves special mention. Mr.E. Rainer's help in processing slides for the presentation of this work is also gratefully acknowledged.

A special word of thanks to my friends and roommates during the course of this study - Mr.S. Joykutty, Mr.S. Natarajan, Dr.J. Rajan, Dr.S. Rajanaryanan, Mr.K. Rangan, Mr.Y.S. Rao, Dr.K. Umesh, Mr.A. Vibhas, Mr. and Mrs.R. Seval, Mr.S. Shivakumar, Mr and Mrs.G. Thiagarajan, Dr.L. Wargo and many others for making life away from school a wonderful experience.

Words cannot express my heartfelt thanks to my parents for their constant encouragement, moral support and financial assistance throughout the course of my education. I also wish to thank my uncle Mr. Sampath Iyengar for his general interest in my progress. My brother Murali, and my sister-in-law Vinita also deserve a special word of thanks for their constant encouragement through the course of my studies. My newly born niece Tara also deserves mention: her cheerfully smiling innocent face has brought added cheer to my day.

Contents

Volume I

List of Tables

List of Figures

List of Variables

$\sigma_1, \sigma_2, \sigma_3$	Principal Stress Components ($\sigma_1 > \sigma_2 > \sigma_3$)
$\epsilon_1, \epsilon_2, \epsilon_3$	Principal Strain Components ($\epsilon_1 > \epsilon_2 > \epsilon_3$)
I_1	First Stress Invarient.
J_2	Second Deviatoric Stress Invarient.
J_3	Third Deviatoric Stress Invarient.
f_c'	Uniaxial Compressive strength (Cylinder).
ϵ_c'	Compressive Strain Corresponding to Uniaxial Compressive Strength
σ_{bc}	Biaxial Compressive Strength
$\{\sigma\}_{x,y,z}$	Stress Tensor in Global Directions.
$\{\epsilon\}_{x,y,z}$	Strain Tensor in Global Directions.
$[D]^{co}$	Concrete Stiffness Tensor in Global Directions.
ξ	Mean Hydrostatic Stress.
ϱ	Deviatoric Stress.
E_{co}	Secant Stiffness Modulus of Concrete
ν_{co}	Secant Poisson's Ratio of Concrete
σ_y,	Yield Stress for Steel.
ϵ_y	Yield Strain for Steel.
E_y	Secant Stiffness for Steel.
$[D]^{cr}$	Crack Stiffness Tensor in local Directions.
$E_{cr}, G_{cr1}, \text{and} G_{cr2}$	Crack Normal and Shear Stiffness Components.
G_f	Fracture energy consumed per unit length of crack.
$[D]^{ts}$	Tension-Stiffening Tensor in the Steel Direction.
E^{ts}	Secant Tension-Stiffening Stiffness Component.
$[D]^{cr-co}$	Plain Cracked Concrete Stiffness Tensor in Global Directions

$[D]^{rc-co}$	Cracked Reinforced Concrete Stiffness Tensor in Global Directions Including Tension-Stiffening.
$[D]^{coupling}$	Coupling Stiffness Tensor in Steel Coordinates.
$[D]^{cr-rcon}$	Cracked Reinforced Concrete Including Tension-Stiffening and Coupling.

Abstract

In this study a post-cracking formulation for the analysis of reinforced concrete structures has been proposed. The secant stiffness formulation considers the following nonlinearities: effects of tension-stiffening in concrete along the reinforcing direction(s), aggregate interlock model with variable shear based on crack confining stresses, and reduction of concrete compressive strength and stiffness after cracking as a function of stresses or strains. The cracking model is capable of simulating the effects of multiple non-orthogonal cracks at a point. This total stiffness formulation treats concrete unloading and reloading along a secant path and is capable of simulating the post-peak strain softening response. The reinforcement is treated as an elasto-plastic material with possible strain-hardening having uniaxial stiffness properties. Perfect bond is assumed between concrete and steel.

The proposed nonlinear inelastic secant stiffness model has been implemented into a special purpose finite element program. Implicit and explicit layering procedures have been incorporated in conjunction with shell elements. These layering procedures enable the geometric modelling of different materials through the cross-section. Procedures to compute the variation in the transverse shear stress through the cross section in the implicit layering procedure have been adopted. The transverse shear stress distribution through the cross-section are better represented with these layering procedures and have been included in the material model. The progress of material nonlinearities through the cross section, such as concrete cracking or crushing, are simulated more effectively with these layering procedures.

The capabilities of this 'three-dimensional' material model have been explored by simulating a number of experimental test specimens subjected to various loadings and comparing the analytical predictions with appropriate test results and design code provisions. These examples include a number of panels and wall specimen subjected to in-plane loads; beams and slab specimens which are subjected to out-of-plane loads and a slab element representing the connection region of flat plate

column joints subjected to a combination of loads. The results of these studies indicate that the analytical model is capable of capturing the dominant deformational response characteristics and predicting the appropriate failure modes.

Chapter 1

Introduction

Reinforced concrete (RC) surface structures find their use in our daily life as components of buildings: panels, walls, slabs and shell roofs; and in more complex structures such as pressure vessels, storage tanks, etc. (figure 1.1). Due to their planar geometry, surface structures carry most of the load through membrane (in-plane) stresses. The complexities associated with the development of rational analytical procedures for such structures, have caused existing design methods to continue in many respects to be based on empirical procedures, which to a large extent depend on experimental data. The rapid development of computational hardware and powerful numerical procedures such as the finite element method have made feasible the investigations of the load-deformational response of RC surface structures into the inelastic domain.

To understand the behavior of RC structural systems under complex loading configurations, a number of experimental and analytical studies have been conducted in recent years. These studies have focused on components of such systems to examine the response nonlinearities under various loads and to explore their influence on load carrying capacities. Under a general state of stress in RC elements, cracks are formed when the tensile stresses or strains reach their cracking value. After cracking, the tensile stresses in concrete diminish rapidly at the crack surfaces and are carried instead by the reinforcement. However, it is expected that concrete between these cracks continues to carry tensile stresses and thus contribute to the stiffness of the cracked RC element. The concrete between the cracks is also capable of sustaining compressive stresses in a direction parallel to the crack surfaces. In such cases, it is expected that the ultimate strength and stiffness of concrete compressive 'struts' would reduce considerably due to the lack of confining effect, stemming from the opening of cracks. After reaching the peak compressive stress

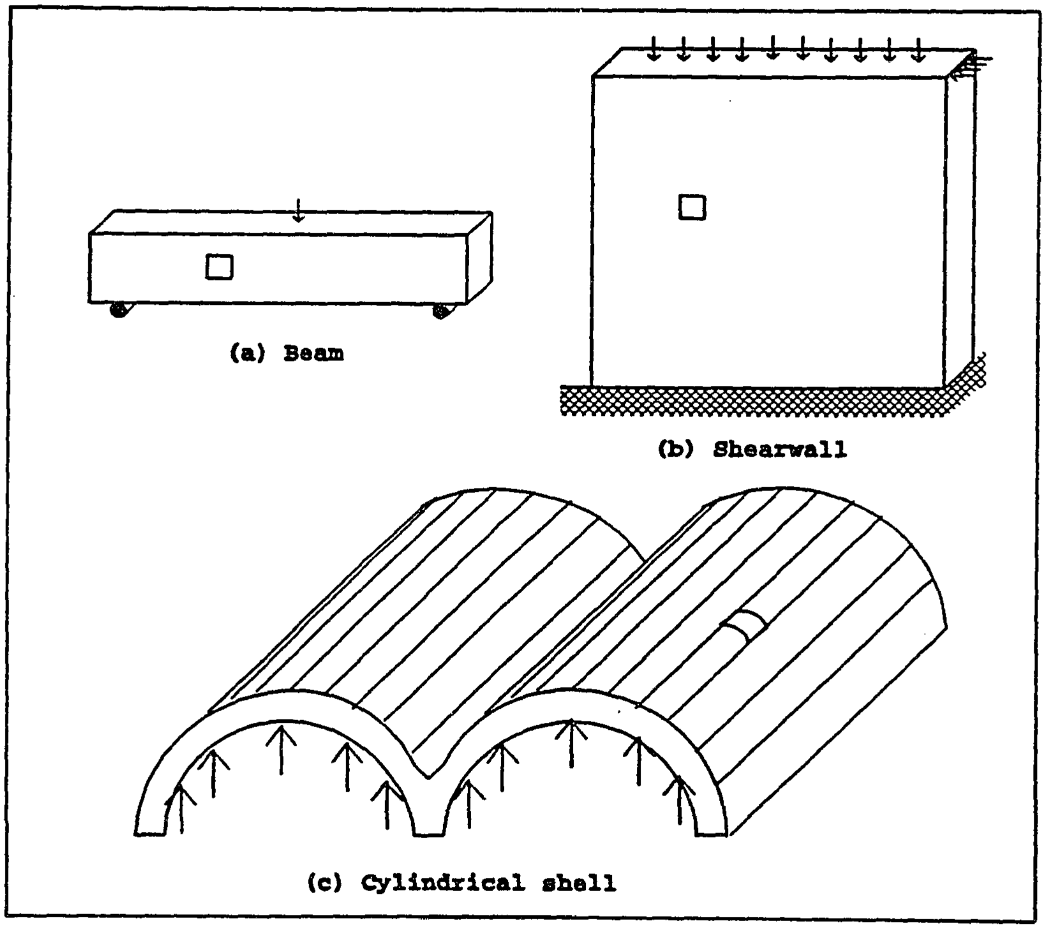


Figure 1.1: Reinforced Concrete Structural Components.

concrete continues to sustain stresses that reduce gradually with increasing strains. This phenomenon is known as post-peak strain softening. Shear stresses are transferred across open cracks through the interaction of rough interfaces. Because of the presence of shear stresses it is possible for new cracks to form at a different angle and previously existing cracks to close. The opening and closing of cracks, crushing of concrete, yielding of reinforcement, and the slippage of the reinforcement (bond slip) may significantly influence the deformational response and the ultimate capacity of RC elements.

Experimental studies have considered numerous parameters such as: types of structural elements (panels, deep beams, shear walls, slabs), concrete strength, type of reinforcement (undeformed and deformed bars, wire mesh), different loading methods (monotonic, proportional, non-proportional, reverse cyclic, etc.), and various loading distribution (distributed edge loads via steel or concrete, point loads etc.) and boundary conditions (simply supported, clamped etc.). Tests of RC panel elements under the combined actions of membrane shear and normal stresses (Vecchio and Collins 1982; Rizkalla and Hwang 1984; Bhide and Collins 1986; H. Shima 1987) have identified the importance of a number of response parameters including the contribution of the concrete between the cracks in the deformational response of the elements up to failure. The experiments reveal that in some cases the directions of the initial cracks do not remain fixed on continued loading, and that the crack shifting behavior is a function of the anisotropic nature of the problem. Similar conclusions have been reached from flexural and torsional tests on RC slab elements (Cardenas and Sozen 1968; Marti et al. 1987a, 1987b). Tests on heavily reinforced panel and slab elements have revealed that the cracking of concrete significantly reduces the compressive strength of concrete in the orthogonal direction. These experimental observations indicate that in order to develop analytical models to simulate the response of RC structures up to failure, many of the aforementioned response characteristics should be considered.

While RC surface structures are primarily intended and designed to carry the applied loads by means of in-plane (surface) stresses, there are cases where they are also subjected to appreciable out-of-plane shear stresses. Examples include

RC offshore containment vessel subjected to ice loads and connection regions of slab-wall-column systems in monolithic RC structures. In RC flat plate structures the transfer of shear and unbalanced bending moments at slab-column connections is a critical design consideration. This aspect of design becomes very important under the action of lateral loads, due to wind gusts or earthquake excitations, when a substantial bending moment has to be transferred across the connection. In addition to the flexural and torsional moments, resulting from the in-plane stresses, large amounts of out-of-plane shear stress are also mobilized.

Due to the large number of parameters that can influence the behavior of RC structures, experimental investigations can only provide a limited amount of information. Further, interaction between the different parameters cannot be isolated and studied. In most structures a prototype test is not economically feasible and the various size effects make direct extrapolation of results of small scale tests to full scale structures questionable.

The high cost of experimental investigations has necessitated the development of numerical models for the analysis of reinforced concrete structures. These models vary in the degree of complexity both in terms of their theoretical basis and their computer implementation. The limitations in their capabilities stem from inability of the constitutive models for concrete and steel to simulate different response characteristics eg. dilatation of concrete prior to crushing, hardening of steel subsequent to yielding etc. Some models are incapable of simulating other kinds of nonlinearities eg. tension-stiffening, bond slip, aggregate interlock, dowel action etc., properly. In a number of investigations (eg. Darwin and Pecknold 1974; Milford and Schnobrich 1984; Channakeshava 1988; Kolleger 1988), development of the constitutive model for reinforced concrete is based on a plane stress assumption, i.e. neglecting the influence of the out-of-plane shear and normal stress components in material model computations.

The type of numerical solution procedures, e.g. incremental or total formulations; shape and type of finite element used, e.g. triangular or quadrilateral elements, shell or solid elements; simple linear or higher order displacement elements; the boundary condition idealizations, e.g. simply supported, clamped etc.; can themselves influence the performance of an analytical model.

The primary aim of the present study is to develop a suitable numerical model, based on a three-dimensional formulation, capable of predicting both the deformational response and the ultimate strength of RC surface structures such as panel elements, shearwalls, slabs etc. For this purpose a special purpose finite element three dimensional finite element program having degenerate shell elements (Ahmad et al. 1970) has been developed. To simulate the out of plane shear stress variations, layering procedures (Figueras and Owen, 1983; Barzegar 1989) has been implemented. In this study, emphasis has been placed on simulating the dominant deformational characteristics and identifying governing modes of failure. The objectives, limitations and scope of the present study are presented in the following section.

1.1 Objectives, Scope and Limitations

- Develop a three-dimensional material model for reinforced concrete considering the following: the behavior of concrete under triaxial stress states, crushing and post-crushing behavior of concrete, cracking and post-cracking behavior of plain concrete, tension-stiffening applied to concrete in the direction of the reinforcement(s) and its coupling effects, compression softening of cracked concrete, aggregate interlock, and yielding of the reinforcement. The simulation of post-cracking aspects of reinforced concrete response has received special attention.

- Implement the proposed material model into a special purpose finite element program consisting of solid and degenerate shell elements.

- Implement layering procedures to consider the variation of material properties and geometry, and treat the development and propagation of cracks and other nonlinearities across reinforced concrete cross-sections.

- Implement procedures to compute the transverse shear stress components through the cross-section in conjunction with the layering procedures for inclusion in the material model.
- Evaluate the performance of the analytical model by analyzing a range of test specimens: panels and shear-walls subjected to in-plane loads, beams and slabs subjected to flexural, torsional and shearing loads by comparing them with the experimental results and ACI design procedures.
- This study is limited to a class of problems where the influence of the out-of-plane normal stress is negligible. Contributions of the out-of-plane shear stresses are included in the material model developed in this study.
- This study is limited to a class of problems where only small strains and displacements need be considered.
- Time dependent effects such as creep and shrinkage and thermal effects have not been included.
- The influence of cyclic loading has not been considered in the material model development.
- Perfect bond is assumed between concrete and steel interface.
- Reinforcement is treated as smeared axial layers neglecting its shear contribution.

1.2 Organization

In chapter 2 of this report a brief review of the different constitutive models for concrete employed in various analytical studies is presented. The constitutive model for uncracked concrete and steel has been selected on this basis. A review of techniques that have been used to model various post-cracking nonlinearities in different analytical investigations is discussed.

In chapter 3 the selected constitutive models for uncracked concrete and steel are presented. The implementation of the material model for concrete is then ratified by comparisons of computed and experimental results.

Chapter 4 of this report deals with the constitutive models for cracked concrete and cracked reinforced concrete. Here the formulations for considering the different post-cracking nonlinearities have been developed.

The implementation of the reinforced concrete model in a special purpose finite element program is discussed in chapter 5. This includes the simulation of concrete, steel, and cracks, the iterative solution algorithm, the numerical integration scheme, the cross-sectional layering techniques, and the procedures employed to test the convergence of the nonlinear algorithm.

The capabilities of the developed models in simulating the different response nonlinearities are explored in chapter 6. A number of experiments on different reinforced concrete structural components vis. panels, walls, slabs are analyzed and the results are compared with the test results.

Conclusions based on this study are then presented in chapter 7. Possible extensions for examination in the future are also identified.

Chapter 2

Literature Review

2.1 Introduction

In order to analytically predict the complete inelastic response of reinforced concrete structures up to failure, the employed model must be capable of properly representing the various nonlinearities present. The time-independent nonlinearities include concrete response under compression up to crushing, post-crushing strain softening behavior, reinforcement yielding and post-yield hardening, cracking of concrete, tensile strain-softening response of plain concrete, stiffness contribution of concrete between cracks referred to as 'tension-stiffening', aggregate interlock, dowel action, and degradation of cracked concrete compressive stiffness and strength and stiffness with lateral tensile strains. Further, when cyclic or reversed cyclic loading is considered, the model should additionally incorporate progressive degradation of material properties with the number of cycles of load application and also capture the energy loss in each unloading and reloading cycle (hysteretic behavior). In this chapter a brief overview of some of the capabilities of analytical models developed for simulating different response behavior of RC elements to various loads is reviewed. These include the various constitutive models employed for simulating steel and plain concrete responses, procedures for simulating cracks, and the post-cracking nonlinearities considered in different studies.

2.2 Constitutive Modelling of Steel

Constitutive modelling of steel reinforcement has been discussed in detail in ASCE (1982). Typically, a uniaxial stress-strain relationship is employed for steel up to yielding. The post-yield behavior is idealized as either perfectly plastic or hardening plastic. The isotropic hardening parameter is usually defined in terms of the stiffness at initial yield. In cyclic loading studies, the Bauschinger's effect has been included through the use of kinematic hardening.

To represent the reinforcement in finite element formulations three different approaches have been considered - (1) Smeared layer, (2) embedded bar, (3) discrete bar elements (figure 2.1).

In the smeared layer approach the steel is distributed across the entire concrete element and is oriented in the direction of the steel bar as shown in figure 2.1 (a). In the embedded formulation the steel bar is located inside the concrete element and displacement compatability between the two materials is enforced at the ends (figure 2.1 (b)). In the discrete element approach steel bar elements are connected explicitly with the surrounding concrete as shown in figure 2.1 (c).

2.3 Constitutive Modelling of Uncracked Concrete

In order to simulate the precracking response of concrete, a number of constitutive models have been proposed in the literature. These can be broadly categorized into the following groups: elasticity based models, plasticity based models, damage mechanics based models and models based on endochronic theory. The ASCE (1982) task committee report, presented an exhaustive survey on the state of the art in reinforced concrete analysis up to that time. This report examined the constitutive models employed for uncracked concrete, the different failure criteria for concrete, the procedures for simulation of cracks, post-cracking nonlinearities considered in different studies, constitutive models for steel, and the type of problems examined using these models. Hence, the emphasis in this survey of literature is on more recent work while a limited number of earlier studies relevant to the present discussion are also included.

2.3.1 Failure Criteria for Concrete

Chen (1982) examined a number of failure criteria employed for concrete and presented the details of the different failure criteria. These include the one parameter envelope (Von Mises criterion), two parameter envelope (Mohr-Coloumb crite-

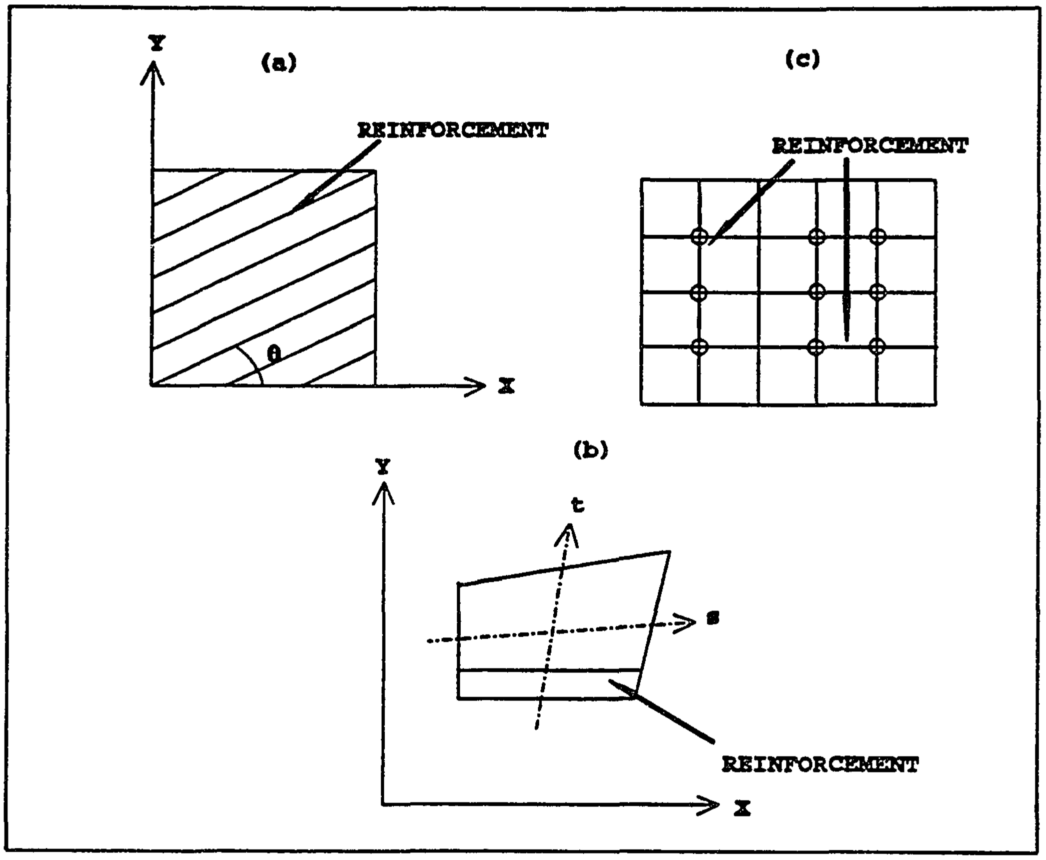


Figure 2.1: Representation of Steel in Finite Element Analysis of Reinforced Concrete (a) Smeared Steel Layer (b) Embedded Steel Element (c) Discrete Bar Element.

rion), Willam-Warnke's three and five parameter envelopes (1975), Ottosen's four parameter envelope (1977), and Hsieh-Ting-Chen four parameter envelope (1979). These models have been examined through correlations with available test data. Chen's study concludes that of these different criteria, the four parameter model proposed by Ottosen (1977), the four parameter model proposed by Hsieh-Ting-Chen (1979) and the three and five parameter models proposed by Willam and Warnke (1975) are the most suitable, capable of properly representing ultimate strengths under general stress states. These models have all the required characteristics of a failure surface vis. smoothness, convexity, symmetry and curved meridians. They also include the one and two parameter envelopes as special cases.

2.3.2 Stress-Strain Response in Concrete

2.3.2.1 Elasticity Based Models

A number of researchers have employed elasticity based models primarily because these are simpler to implement. In these models the nonlinear behavior of concrete is represented by appropriately changing the secant modulus (total formulation) or tangent modulus (incremental formulation). The models by Liu et al. (1972), Darwin and Pecknold (1974), Kotsovos and Newman (1978), Ottosen(1979), Bazant and Tsubaki (1979), Gerstle (1981), Stankowski and Gerstle (1985), etc. are the most notable.

Incremental Formulations

In an incremental formulation the incremental stresses are related to the incremental strains as $\{\Delta\sigma\} = [D]\{\Delta\epsilon\}$, where [D] is the tangent stiffness matrix. The material parameters used in this procedure depend on the current incremental stress or strain and the stress or strain history up to that point. In general, incremental elasticity based models have the following characteristics:

- This approach is an incrementally linear procedure having path dependent characteristics present in it.

- They are simple to implement and can generate the stress-strain behavior of concrete accurately if a broad base of data is available.
- Their applicability is restricted to states of stresses for which the material parameters have been calibrated.
- Loading, unloading and neutral loading conditions are not properly defined.
- They do not treat the post-peak strain softening behavior properly.

A brief review of some incremental elasticity based models follows. Liu et al. (1972) proposed an incremental orthotropic model that accounts for stress induced anisotropy. The material parameters are modified based on total strains. The principal stresses are expressed as closed form expressions in terms of the total strains and the ratio of the principal stresses. The axis of orthotropy is assumed to coincide with the current principal stress direction. Milford and Schnobrich (1984) have employed this model in their study of reinforced concrete cooling towers.

Darwin and Pecknold (1974) proposed an incremental orthotropic model which is based on the concept of equivalent uniaxial strains. Here the effects of biaxial stresses on internal damage in concrete are represented by equivalent uniaxial stress-strain curves in each principal stress direction. The equivalent uniaxial strains are not real strains and do not transform under axis rotations as do true strains and are significant only as a measure on which to base the variations in material properties. The principal stress axis may rotate and need not coincide with the strain direction. The equivalent uniaxial strains are accumulated in the stress directions. This is done without taking into account the axis rotation. They have used this model for the analysis of beams subjected to static and cyclic loadings. Noguchi (1985) has employed this model for concrete in his study of beam-column joints.

The orthotropic models of Liu et al. (1972) and Darwin and Pecknold (1974) have been strongly criticized by Bazant (1979) on both physical and theoretical grounds.

Stankowski and Gerstle (1985) have proposed a simple hypoelastic formulation for concrete based on data from tests under complex load histories. This incremental formulation relates the octahedral stresses to octahedral strains through the bulk and shear modulii. There is also an interaction between hydrostatic and deviatoric stress components through coupling modulii obtained from experiments. This model has been validated for special axisymmetric loading cases. This formulation is a modification of a model proposed by Gerstle (1981) where the coupling between the hydrostatic and deviatoric stress components was not considered.

Total Formulation

In a total formulation the total stresses and total strains are related through the secant stiffness matrix [D] as $\{\sigma\} = [D]\{\epsilon\}$. The secant material modulii of concrete are obtained directly from experimental data. In general, total elasticity based models have the following characteristics:

- The use of this approach implies path independent behavior, while concrete exhibits path dependency.
- They are simple to implement and can generate the stress-strain behavior of concrete accurately if a broad base of data is available.
- Their applicability is restricted to states of stresses for which the material parameters have been calibrated.
- Loading, unloading and neutral loading conditions are not properly defined.
- Strain softening response can be simulated with relative ease.

Kotsovos and Newman (1978) proposed a constitutive model for concrete characterized by a total formulation. The two secant material properties, bulk modulus and shear modulus, are evaluated based on curve fitting the data related to concrete compressive strength. The octahedral stresses are related to the octahedral strains through these modulii. A coupling stress which is a function of both the octahedral stress and the deviatoric stress is also defined and employed in computing the octahedral strains.

Ottosen (1979) has developed a secant nonlinear elastic model for concrete. The Young's modulus and Poisson's ratio are modified based on a nonlinearity index. This index relates the current most compressive stress to its ultimate value. The model has been compared with biaxial and triaxial test data for both proportional and non-proportional loading situations and has performed reasonably well. Ottosen has used this formulation in the study of panels, beams and pull-out test specimens.

Bazant and Tsubaki (1979) developed a total strain formulation which includes path dependent parameters. The expressions used are algebraic and independent of loading surfaces or intrinsic time functions. The model has been compared with test data and performs as well as endochronic theories and plastic fracturing models. However, the formulation is not explicit and needs to be modified into its incremental form for implementation and thus is complex.

2.3.2.2 Plasticity Based Models

Plasticity based models have become popular because they are relatively simple to implement, do not depend on a large number of parameters for calibration, and have a sound mathematical basis in their formulation. In these models it is possible to compute the strains due to elastic behavior (fully recoverable) and those due to plasticity (irrecoverable) separately. The recoverable strains are treated within the framework of elasticity theory while the irrecoverable plastic strains are computed based on the theory of plasticity. These models can be further classified on the basis of their hardening rules, as perfectly plastic and hardening plastic models. Hardening plasticity requires two surfaces - the loading surface and the failure surface. Typically the behavior of concrete is assumed to be elastic up to 30% of its uniaxial compressive strength. A loading surface is assumed at this level which is usually assumed to be similar in shape to the failure surface. This loading surface expands with increasing loads based on hardening laws (isotropic, kinematic, mixed) until it meets the failure surface. The components of the plastic strain increments are computed based on assumed flow rules (associated, non-associated).

Chen (1982) has discussed the plasticity based formulations used in RC analysis prior to 1982 in detail. Some of the general characteristics of plasticity based models are:

- They guarantee a stable and unique description of the material law when the normality rule and convexity requirements (Drucker's postulates) are satisfied.
- Loading, unloading and neutral loading conditions are properly defined.
- They account for stress history dependent behavior.
- Residual strains can be computed.
- They do not account for any stiffness degradation.
- Numerical difficulties are encountered when simulating post-peak strain- softening behavior (Glemberg and Samualsson 1983 is an exception).
- They predict an unusually high volume expansion for large hydrostatic compressive stresses and thus represent dilatation poorly.

A brief survey of some of the more recent plasticity based models is presented herein.

Glemberg and Samualsson (1983) have proposed a concrete constitutive model based on the theory of plasticity. This model employs an isotropic strain hardening law up to the ultimate strength. Beyond ultimate strength, the strain softening behavior is also included using exponential functions which define the shrinking of the failure envelope. In their formulation negative definite stiffness tensors are permitted.

Han and Chen (1985) have proposed a non-associated plasticity model. The hardening characteristics of this model account for its ductility in compression and brittleness in tension. The shape function used to define the loading surface is determined such that for triaxial tension no hardening is present, thus representing brittle failure. This shape function is designed such that the hardening zone increases with hydrostatic compression. To adequately describe volume contraction and dilatancy, a non-associated plastic flow rule has been employed.

Hu and Schnobrich (1987) have employed a plasticity based model in their study. The formulation includes an isotropic hardening law and a multiple yield criterion based on the stress region. Both associated and non-associated flow rules have been employed along with an equivalent uniaxial stress-strain law.

Torrent et al. (1987) have employed a work hardening plasticity model and failure of concrete under multiaxial stresses. The initial yield surface (loading surface) is closed, while the failure surface remains open along the negative hydrostatic axis, similar to the Ottosen failure criterion (1977). The model shows good agreement with test data.

Channakeshava (1987) has used an elasto-plastic model. This model incorporates isotropic hardening, a segmented equivalent uniaxial stress-strain law, and a modified Mises envelope for its loading and failure surface. In this study it is assumed that plastic stress redistribution follows fracture stress redistribution. An associated flow rule is used to evaluate the plastic strains.

Frantzeskakis (1987) has proposed a model based on a non-associated plasticity flow rule. The yield surface is defined as a Drucker-Prager function. The plastic potential employed to calculate the plastic strains is also a Drucker-Prager function.

Bounding Surface Plasticity Models

Bounding surface plasticity models first evolved to consider the degradation of material properties due to cyclic loading in metals. This approach has since been used to describe the behavior of concrete. A brief review of some of the recent models using this approach is presented here.

Fardis et al. (1983) have used a bounding surface model in stress space proposed by Dafalias and Popov for metals. The bounding surface encloses the current stress point for a given state of stress and strain. This surface is a function of the stress state and the maximum principal compressive strain experienced by the material. The plastic strains are functions of the distance between the current stress point and the bounding surface measured along the instantaneous loading direction.

Buyukozturk and Chen (1985) have proposed a rate independent constitutive model for multiaxial cyclic loading. The concrete is assumed to experience a continuous damage process under load histories. A damage dependent bounding surface in stress space is used to predict the strength and deformational characteristics of the gross material under general loading paths. The size of the bounding surface reduces with increase in damage and the functional dependence of the material modulii on stress/damage gives a realistic picture. Cyclic degradation of properties, post-failure strain-softening and shear compaction dilatancy are also included in this model.

Eberhardsteiner et al. (1987) have proposed a bounding surface plasticity model. In the pre-failure region, the bounding surface is fixed in the stress space. The direction of the plastic flow is determined on the basis of experimental test data. The loading criterion employed by Stankowski and Gerstle (1985) is used in this study.

2.3.2.3 Plastic Fracturing Theory

While the treatment of concrete deformation response in pre-failure stress states has been well established through plasticity based models, the constitutive aspects of concrete in the post-failure region has met with some difficulty. For a given uniaxial stress two strain values can be identified from the stress-strain relationship for concrete, one prior to failure and one in the post-failure softening region. Thus, the strains are not uniquely defined for a given general stress state. Moreover, strain-softening response in concrete is due to microcracking which, in contrast to plastic phenomena, is accompanied by a decrease of elastic modulii. Models formulated in strain space have been proposed to consider the softening of the stiffness modulus (e.g. Dougill 1976).

Bazant and Kim (1979) proposed a constitutive model for concrete based on plastic fracture. The inelastic behavior of concrete is assumed to consist of both plastic and fracturing deformations. The plastic strains are irreversible and do not cause a change in the stiffness modulus. The fracture deformations, on the other hand, are reversible and cause reduction of the stiffness modulus as a result of micro-cracking. The plastic strain increments are based on loading surfaces and flow rules formulated in stress space. Similarly the fracture process is described by using a decrement in stresses along with a fracturing surface described in strain space. One important advantage of the plastic fracturing theory is that, in contrast to incremental plasticity, it gives inelastic responses for stress increments tangential to the loading surface (neutral loading), whereas the classical plasticity theory gives a perfectly elastic response for such load increments, which is not true for concrete.

Han and Chen (1985) have examined this model and found it capable of describing the nonlinear behavior of concrete up to ultimate. However, the use of two loading surfaces makes it very complicated to define criteria for loading, unloading, and reloading of the material. Han and Chen (1985) have employed an internal volumetric work criteria to describe the loading/unloading state of the concrete in their studies. Yamaguchi and Nomura (1985) have employed the plastic fracture model in a study of shear walls subjected to cyclic loads. The results obtained in this study compared well with the experimental response.

2.3.2.4 Endochronic Theory Based Models

Endochronic theories for modelling concrete were first employed by Bazant and Bhat (1976). The central concept in this theory is that of intrinsic time. The increments of intrinsic time, which are non-negative, depend on the total strain increment. The size of the intrinsic time increments controls the inelastic strain increments. Thus the endochronic theory represents a special type of viscoplasticity in which the plastic rate coefficient depends not only on the stress and strain but also the strain rate. With their implementation of this formulation Bazant and Bhat (1976) have shown that a wide range of phenomena can be modeled, e.g., strain softening region, volume dilatation, hydrostatic pressure sensitivity and the strain rate effect. But the number of parameters required to describe this formulation is large compared to elasticity or plasticity based models. A number of refinements to the original formulation have been proposed by Bazant (1980) including a loading criteria. While incremental elasticity and plasticity based models are incrementally linear, the endochronic theory based models are incrementally nonlinear. However, with the use of loading and unloading criteria the performance of endochronic theory based models has been found to be no better than models using classical plasticity based formulations.

2.3.2.5 Damage Mechanics Based Models

Damage mechanics based models have been proposed in the last few years to consider the microcracking damage taking place even before failure. These models are capable of computing the degradation of stiffness with increasing damage that is not considered in classical plasticity. A few models employing this strategy are presented herein.

Ishikawa, Yoshikawa and Tanabe (1985) have proposed a constitutive model for concrete in terms of a damage tensor. The damage tensor is defined in terms of a 3-D damage field where the damage increase in one direction is used to affect the damage rate in the other two directions. From this a 3-D damage strain tensor is defined to allow for a stress-analysis as an initial strain problem. The model is compared to the model of Kupfer et al. (1973). The stress and strain histories in uniaxial cyclic loading are also considered and reasonably simple expressions are obtained.

Ramtani et. al. (1989) have developed an anisotropic damage model. The anisotropic degradation of the material properties is represented by a second order damage tensor. Inelastic strains are included in the stress-strain relations by coupling the damage to the strains. Evolution laws are associated with each damage variable.

Oliver et al. (1990) have proposed an isotropic damage mechanics based model to simulate the behavior of concrete under different stress-strain conditions with special attention to fracture and strain localization under tension. The model employs a damage criterion formulated in strain space or undamaged stress space.An equivalent strain norm is used to evaluate loading,unloading and reloading. Evolution laws for the damage variables and the damage threshold have been proposed. Comparisons between this model and the conventional anisotropic models using simple tension tests and tension-shear test data have also been made. The model has a simple formulation, is easy to implement, has an explicit integration scheme, and is based on a consistent constitutive theory free from empirical factors.

Borderie et. al. (1990) have proposed a damage mechanics model capable of representing crack opening/closing. This model uses damage coefficients which are internal state variables along with variables describing the state of the crack. The constitutive equations employed are derived based on a thermodynamic irreversible process. The performance of this model has been examined in a two dimensional finite element study of tests on beams subjected to cyclic loading.

2.3.2.6 Comparative Studies of Constitutive Models

Inorder to evaluate the capabilities of the different constitutive models used for concrete it is necessary to make a comparison of these models using some bench mark problems. A number of such studies have been undertaken and a brief evaluation from two such investigations are presented herein.

Eberhardsteiner et. al. (1987) have examined some of the constitutive models employed for concrete. In their study four models were considered: a nonlinear elasticity based model due to Kotsovos and Newman (1978); a Hypoelasticity based model due to Gerstle and Stankowski (1985); an elasto- plastic model due to Han and Chen (1985); and bounding surface plasticity model proposed by Eberhardsteiner et al. (1987).

The performance of these models in non-proportional and non-monotonic loading cases has been evaluated. The constitutive model proposed by Kotsovos and Newman (1978) was found to be inadequate as it lacked parameters dependent on loading histories. The other models were found to be able to predict the non-proportional loading paths with some sucess, especially the bounding surface plasticity model due to Eberhardsteiner et. al. (1987).

Torrenti and Djebri (1990) have also presented a comparative study of some other constitutive models employed for concrete. For this study, elasticity based models of Ottosen (1979) and Ahmad and Shah (1986); continuum damage model of Mazars (1989); hypoelastic model of Gerstle (1981); orthotropic model of Torrenti (1987); plasticity based models of Frantzeskakis(1987) and Fardis (1983) have been considered. The implementation of each model in their study has been undertaken without special tuning for particular problems. These models were examined for proportional and non-proportional as well as in two and three dimensional loading conditions. The predicted stress-strain responses using all the models in two dimensional proportional loading cases were found to correlate well with the experimental data of Kupfer et al. (1969). In the case of non-proportional loading, the model proposed by Gerstle (1981) and that of Fardis (1983) were found to be inadequate; the most compressive principal stress direction was found to change compared with the experimental data resulting in lower failure loads. In three dimensional loading cases the incremental model proposed by Gerstle (1981) was satisfactory in reproducing the most compressive principal strains in tests reported by Balmer (1949). But the strains in the other two directions were smaller than their test values. The other models employed were found to underestimate the straining even in the most compressive strain directions. However, they conclude that for most macro analysis any of the models considered here would perform adequately.

2.4 Crack Simulation and Post-Cracking Nonlinearities

2.4.1 Crack Simulation

The representation of cracks in the finite element analysis of concrete structures has been accomplished in two different ways: discrete cracks (Ngo and Scordelis 1967) and smeared cracks (Rashid 1968). In discrete crack modelling the nodes along which the crack can propagate are double or four noded, or even eight noded and are uncoupled when the stresses at a particular node violates the cracking criteria (figure 2.2 (a) and (b)).

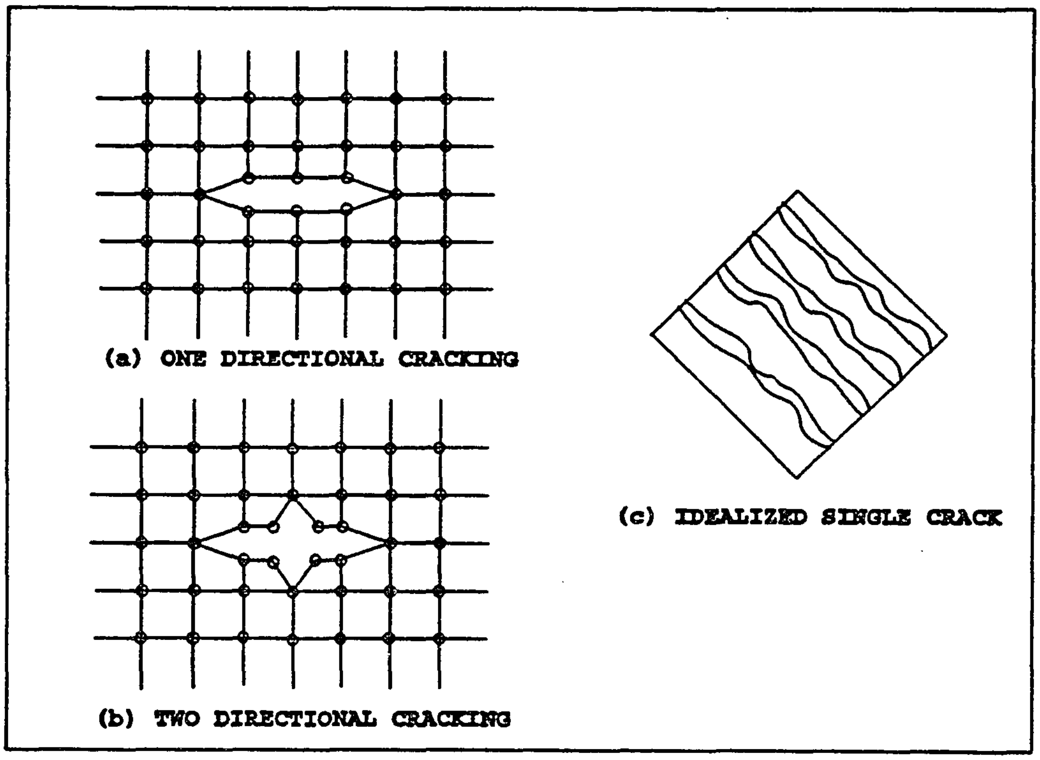


Figure 2.2: Representation of Cracks in Finite Element Analysis of Concrete (a) and (b) Discrete Cracks (c) Smeared Cracks.

The use of discrete cracking representations has received only limited acceptance due to the difficulty involved in redefining the mesh topology after the crack formation. Further, the uncoupling of the nodal points destroys the banded structure of the global stiffness matrix, greatly increasing the computational effort required. The inclusion of post-cracking nonlinearities through coupling springs at the 'cracked nodes' requires additional parameters which are difficult to obtain from experimental data. Therefore discrete crack based models are not considered any further in this study. In the smeared crack approach the points which 'crack' are assumed to have a reduced stiffness in the direction normal to the crack plane (figure 2.2 (c)). A number of studies have used this approach stemming from its ease of implementation. Additionally, post-cracking nonlinearities are included in a simple manner. A brief review of some of the smeared crack models is presented next.

Some studies have treated the concrete as isotropic prior to cracking and orthotropic once it cracks. The axes of orthotropy are kept normal and tangential to the crack plane (e.g., Darwin and Pecknold 1974). The crack direction is then considered 'fixed' during subsequent loading stages. The stiffness in the two directions are computed independently and the Poisson's effect is assumed lost. Most formulations include a shear stiffness component which is a fraction of its value prior to cracking. Initiation of new cracks is controlled by specified parameters (threshold angle between any two cracks, etc.). In the presence of anisotropic reinforcement, experimental observations indicate that the crack directions change with the increased load application. In such cases, this modelling approach has been found to be unduly stiff in its predictions and overestimates the ultimate strength of test specimens considerably (Barzegar 1988a).

In an attempt to incorporate the anisotropy induced by cracking, and the effect of anisotropic reinforcement, in some studies (Cope et al. 1981, Gupta and Akbar 1984; Milford and Schnobrich 1984; Noguchi 1985; Maestrini and Gupta 1987; Balakrishnan and Murray 1988; Hu and Schnobrich 1988; Kolleger 1988, Ola and Ottosen 1990) the crack direction has been allowed to 'rotate' with subsequent loading. The crack direction is assumed to be perpendicular to the maximum principal tensile strain direction upon further loading. In this approach previously formed cracks are ignored, thus the formulation violates the tensorial invariance properties (Bazant 1983).

Another approach in the context of smeared crack modelling considers the cracked concrete to be composed of intact (or uncracked) concrete and microcracks, idealized as springs in series (de Borst and Nauta 1985; Barzegar and Schnobrich 1986; Channakeshava 1987; Rots 1985b, 1988). Each crack interface has a stiffness normal to the crack plane as well as a shear stiffness. This concept permits further non-orthogonal cracks to develop without neglecting the effect of previously formed damage planes. Additionally, it simplifies the monitoring of the crack state namely the crack opening, closing, and reopening.

2.4.2 Post-Cracking Nonlinearities

2.4.2.1 Strain-Softening

The formation and propagation of cracks in plain concrete is a gradual process accompanied by a complete loss of strength and stiffness perpendicular to the crack direction. It has been argued (Bazant and Oh 1983) that a strength based criterion for crack propagation is not objective with respect to finite element mesh refinement. Thus if the mesh were refined in front of a given crack different load increments would be required to advance the crack by a unit length. If such a mesh refinement were continued it would imply crack propagation without any load increment at the limit.

To resolve these issues, energy based criteria employed in fracture mechanics have been advanced (Dougill, 1976; Hillerborg et al., 1976; Bazant and Cedolin 1979, 1983; Willam et al., 1984). In this approach the stiffness of concrete normal to the crack plane, i.e. microcrack stiffness, is gradually reduced with increased straining in that direction. The fracture energy consumed to advance the crack by one unit length is assumed to be a material property. The fracture energy dissipation, which controls the softening process, is a function of the crack band width associated with the damaged sampling point. Thus as the deformation grows the elements around the cracked point begin to unload while the cracked point undergoes strain-softening. Eventually the crack is fully open and the stresses across the crack drop to zero. This process of strain softening in plain concrete has been discussed in great detail by Willam et. al (1984). A number of studies have focused on capturing the stiffness properties of concrete after cracking. Recent studies have shown (Petersson 1980; Dodds et. al. 1982; Bazant and Oh 1983; Willam et. al. 1984; Rots et. al. 1985, 1988) that the shape of the stress-strain curve after cracking has a significant influence on the propagation of cracks and the post cracking deformational response.

Recently De Borst (1990) has proposed a constitutive model based on Cosserat theory. In this approach the governing field equations include rotational degrees of freedom in addition to the conventional degrees of freedom. This Cosserat continuum model warrants convergence of load deflection curves to a unique solution upon mesh refinement and a finite width of the localized zone. This approach does not employ the 'characteristic length factor' which is required in conventional models in order to overcome mesh dependencies when softening is present. The performance of this model has been examined using biaxial tests composed of strain-softening elasto-plastic materials.

2.4.2.2 Tension-Stiffening

In reinforced concrete structures, the stiffness contribution of concrete between adjacent cracks, known as 'tension stiffening', has been simulated in a number of studies (ASCE 1982; Milford and Schnobrich 1984; Chang et. al. 1987; Barzegar and Schnobrich 1988b). Figure 2.3 shows the variation of stresses in a cracked RC element. The assumption of 'perfect bond', i.e., the steel and concrete fiber adjacent to the steel have the same strain, is inherent to all models considered in this section. This greatly simplifies the implementation of these models but restricts their range of application in, e.g.large diameter bars in monotonic loading and cyclic loading.

In some models (Milford and Schnobrich 1984; Chang et al. 1987; Barzegar and Schnobrich 1988b) tension-stiffening is considered to be smeared on the concrete in a direction normal to the crack. Rots (1988) treats this stiffness contribution as part of the concrete and artificially increases the fracture energy of plain concrete

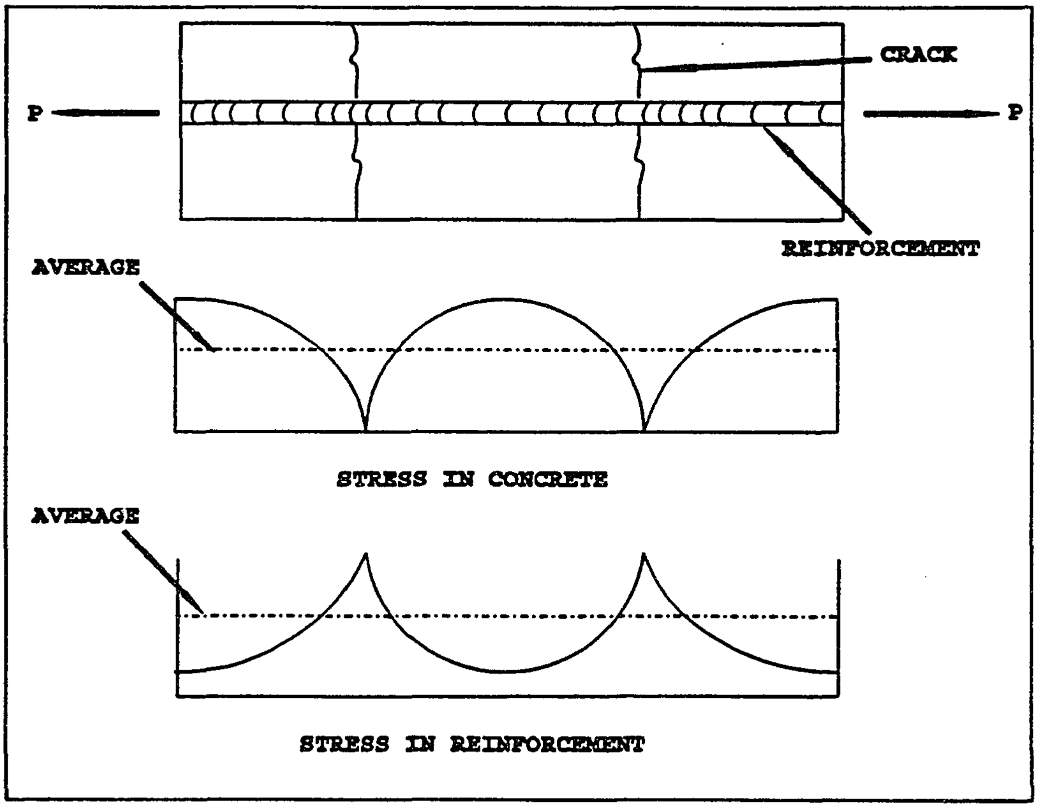


Figure 2.3: Stress Distribution in Cracked Reinforced Concrete.

to accommodate the increased stiffness normal to the crack plane. The disadvantage of treating the stiffness normal to the crack as part of the concrete is in the difficulty experienced in accurately estimating the amount of energy dissipated when cracks are skewed with respect to the reinforcement.

In another formulation (Gilbert and Warner 1977) the concrete contribution between cracks is lumped and considered along the reinforcement direction, assuming the concrete to have lost its stiffness normal to the crack plane. Lumping of the tension-stiffening as part of steel resolves the problem of estimating the termination of tension-stiffening. But the state of stress in concrete is not accurately represented as the tensile stresses are underestimated. Kolleger (1988), treats the tension-stiffening as part of the concrete evaluated in the steel direction and transformed to the principal strain direction ('rotating crack model'). Barzegar and Ramaswamy (1990) treat the tension stiffening as part of the concrete and evaluate it in the steel direction. Such a treatment of tension-stiffening provides a more accurate estimation of the stiffness contribution of concrete after cracking.

2.4.2.3 Bond-Slip

The increased application of loads causes the damage zone to expand and a considerable amount of slippage between steel and concrete is possible. This is especially severe under cyclic loading and when large diameter bars are used. Some studies (Floegl and Mang 1982; Rots 1985; Noguchi 1985; Balakrishnan and Murray 1988; Channakeshava 1987; Gupta and Maesterini 1987; Keuser and Mehlhorn 1987; Yashikawa and Tanabe 1986a, 1986b) have developed tension stiffening criterion on the basis of allowing slippage. This approach requires that the steel be modeled as a discrete bar. The interface between the steel and concrete is double noded making it a computationally expensive procedure. The linkage between the steel and concrete at each point is described using springs, whose properties are described by explicit bond shear stress-shear slip assumed for this purpose. However, Channakeshava (1987) has considered bond slip for smeared steel layers. Elwi and Hrudy (1989) have described a method of incorporating bond slip in embedded steel formulations.

2.4.2.4 Aggregate Interlock and Dowel Action

Shear stresses developing on a crack surface cause the principal stress directions to shift during subsequent loading. In a number of previous studies, a fixed fraction of the uncracked concrete shear stiffness has been retained for the shear stiffness of cracked concrete (Darwin and Pecknold 1974; ASCE 1982; Milford and Schnobrich 1984; Gupta and Akbar 1984; Gupta and Maestrini 1987; Barzegar and Schnobrich 1986). Some investigations have considered varying the shear stiffness as a function of the crack normal strains (Al Mahaidi 1979; Balakrishnan and Murray 1986 and 1988; Al-Manseer and Phillips 1987); Razaqpur 1988). Kolleger (1988) has used an explicit expression to model the dowel action of the bars. Some analytical studies have considered the influence of the confinement provided by the reinforcement on the sliding action of the crack surfaces (Bazant and Gambarova 1980; Li Baolu et al. 1989; Yashikawa, Wu and Tsubaki 1989) and the crack dilatation due to this sliding. There is considerable experimental evidence that such an interaction can be significant in shear critical problems. This coupling action (shear-dilatancy) should not be neglected when cyclic loading is considered (Pruijssers 1988).

2.4.2.5 Compression Softening of Cracked Concrete

Experimental studies (Vecchio and Collins 1982, 1986) on RC panels subjected to pure shear have indicated that the presence of cracks reduces both the compressive stiffness and strength parallel to the crack planes. They have developed an empirical relationship relating the tensile strains normal to the crack plane to the ultimate strength of the concrete compressive strut between those cracks. Recently other researchers (Kolleger 1988; Dyngland 1989; Shirai and Noguchi 1989) have reached the same conclusion that the strength of the compressive strut is reduced due to the lateral straining in cracked concrete. However, they conclude that the level of strength and stiffness reduction is no more than 20 % of the uniaxial compressive strength and that Vecchio and Collins overestimate this reduction.

2.5 Summary

The distribution of reinforcement in a RC structural component is governed by the nature of forces resisted by them. In compression elements like columns, a few reinforcing bars are employed. While in slabs a large number of small diameter bars are placed across the section to resist flexural and torsional loads. Thus the finite element representation of reinforcement is also governed by the structural element being simulated. Embedded or discrete bars are used in simulating column reinforcement. A smeared layer is more representative of the reinforcement placed in slabs. In the present study the focus is on surface structural elements such as slabs and so a smeared layer procedure is adopted for the simulation of steel. The constitutive aspects of steel modelling considered in this study are discussed in chapter 3.

As discussed in section 2.3 the constitutive modelling of concrete for use in nonlinear analysis of RC structures must have the capability to simulate a number of nonlinearities. While incremental formulations (both elasticity and plasticity based) are in usually able to simulate the response of concrete up to the ultimate load adequately, most of these are incapable of simulating the post-peak strain softening due to numerical difficulties. Thus the stress redistribution and mobilization of alternate load paths are not well defined when these formulations are used in nonlinear analysis of RC structures. However, these models, especially the plasticity based models are capable of capturing the unloading and reloading response and the response under non-monotonic loadings adequately.

While models based on plastic fracturing, endochronic theories and damage mechanics are appealing because these models are capable of capturing most of the observed nonlinearities in concrete, they require a number of parameters to describe them completely. The scatter observed in obtaining even uniaxial properties of concrete e.g. tensile and compressive strength of concrete, makes it very difficult to obtain the various other parameters needed by these models to describe the nonlinear response of concrete.

The major emphasis in the present study is to describe the post-cracking aspects of concrete nonlinearities and thus numerical simplicity of the uncracked concrete modelling has been given precedence. However, the need to adequately describe the behavior of concrete up to and beyond the ultimate load has not been compromised. The constraints placed by these criterion have been adequately satisfied by the Ottosen (1979) nonlinear elastic formulation based on a total formulation. The details of the Ottosen (1979) total secant formulation are presented in chapter 3.

Most analytical studies consider the representation of cracks to be 'smeared' over the sample point. This treatment is simple and the computational requirements are limited. This approach has been found to be suitable for load displacement type 'macro' analysis and the more detailed local stress-strain type 'micro' analysis. It has also been found to represent the localization of cracks with a reasonable level of accuracy (Rots 1988). The treatment of crack rotation via the 'rotating crack' model has come under some criticism because it violates tensorial invariance requirements (Bazant 1983). The more recent multiple non-orthogonal cracking formulation accounts for previously formed damage planes and for the states of these damage planes vis. crack opening, closing and reopening. For this reason a smeared crack formulation based on the later approach is used in this study.

The treatment of post-cracking nonlinearities such as tension-stiffening, bond slip, aggregate interlock, compression strut softening and dowel action have been undertaken in different ways in the analytical studies. However, the formulations employed for incorporating these nonlinearities have some inherent difficulties as discussed. In the present study a new post-cracking formulation for tension-stiffening, aggregate interlock, and softening of the compressive strut in cracked concrete have been proposed. The details of the proposed crack formulation and the treatment of post-cracking nonlinearities are presented in chapter 4.

Chapter 3

Constitutive Modelling

3.1 Introduction

In this chapter the constitutive models for concrete and steel employed in this study are presented. The material model for concrete consists of the uncracked concrete stress-strain relationship up to crushing including the post-peak strain softening behavior. Criteria employed to predict cracking or crushing failure are also presented. Treatment of post-cracking phenomena in concrete is addressed separately in chapter 4. The stress-strain relationship for steel up to yield and the post-yield strain-hardening behavior are also described in this chapter.

3.2 Constitutive Modelling of Uncracked Concrete

3.2.1 Failure Criterion for Concrete

In formulating a failure criteria for concrete under combined states of stresses, a proper definition of failure must first be made. Criteria such as yielding, load carrying capacity, initiation of cracking and extent of deformation have been used to define failure. In this study failure means the stresses cannot continue to increase beyond this point (figure 3.1). Generally, concrete failure can be divided into two types: cracking and crushing. These are characterized by brittleness and ductility respectively. Tensile failures are represented by the formation of a number of cracks and the eventual loss of tensile strength and stiffness normal to the crack plane. While in compression a number of microcracks appear with loss of strength in all directions. In this study a four parameter strength envelope proposed by Ottosen

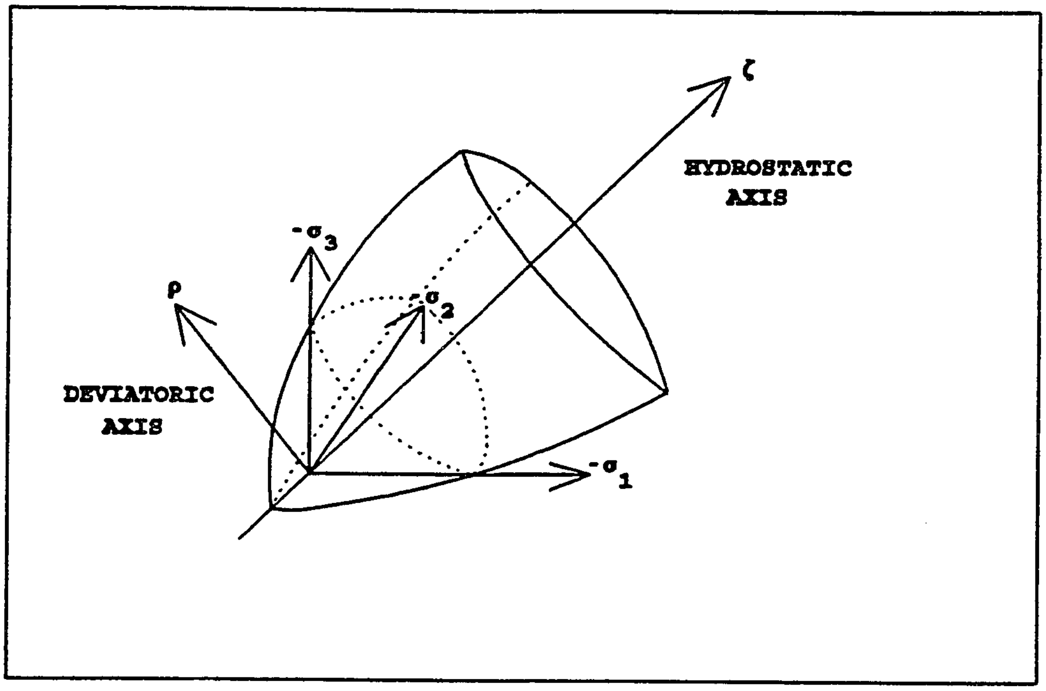


Figure 3.1: Failure Envelope for Concrete Based on Stresses.

(1977) has been employed. This strength envelope is expressed in equation 3.1 as:

$$A\frac{J_2}{f_c'^2} + \lambda\frac{\sqrt{J_2}}{f_c'} + B\frac{I_1}{f_c'} - 1 = 0 \tag{3.1}$$

Where I_1 and J_2 are the first stress invariant and the second deviatoric stress invariant respectively. f_c' is the uniaxial compressive strength of concrete. A and B are parameters, to be determined, and λ is expressed as (Ottosen, 1977):

$$\lambda = K_1 \cos\{\frac{1}{3}\arccos[K_2 \cos 3\theta]\} \qquad \cos 3\theta \geq 0 \tag{3.2}$$

$$\lambda = K_1 \cos\{\frac{\pi}{3} - \frac{1}{3}\arccos[-K_2 \cos 3\theta]\} \qquad \cos 3\theta \leq 0 \tag{3.3}$$

where K_1 and K_2 are parameters. The four parameters A, B, K_1, and K_2 needed to describe the failure surface are evaluated using the following experimental data:

- Uniaxial compressive strength f_c'.
- Biaxial compressive strength of concrete, $\sigma_{bc} = 1.16f_c'$ corresponding to tests conducted by Kupfer (1969, 1973).

- Uniaxial tensile strength f_t'

- The method of least squares has been employed to obtain the best fit of the compressive meridian for $\frac{\xi}{f_c'} \geq -5.0$ to the triaxial test results of Balmer (1949) and Richart (1928) (figure 3.2). The compressive meridian is made to pass through $(\xi, \varrho) = (-50.0\text{MPa}, 43.61\text{MPa})$.

Thus, the values of the four parameters are obtained for any concrete based on the uniaxial properties of the concrete used, *vis.* f_c' and f_t'. Figure 3.2 shows comparisons of the Ottosen strength envelope with the triaxial and biaxial test data.

The characteristics of the Ottosen failure surface represented by equation 3.1 are:

- Only four parameters are needed to describe the envelope,

- Use of invariants eliminates the need for computing principal stresses to determine failure,

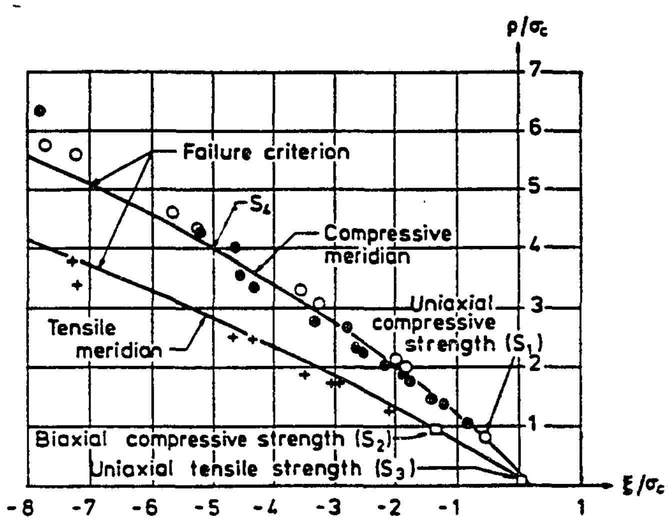
- The surface is smooth and convex except at the vertex,

- The meridians are parabolic and open in the direction of the negative hydrostatic axis,

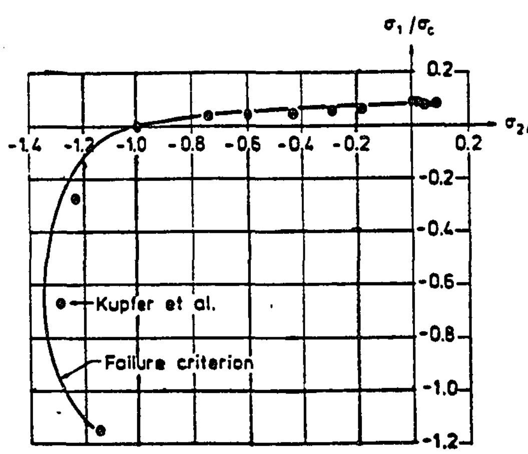
- The trace of the failure surface on the deviatoric plane changes from a triangular shape (hydrostatic tension) to a nearly circular one with increase in hydrostatic compression,

- It contains several earlier failure criteria as special cases. The Drucker-Prager criterion is obtained by setting A=0 and keeping λ constant, while the Von Mises criterion is obtained setting A=B=0, keeping λ constant.

In this study A, B, K_1 and K_2 are evaluated from the uniaxial compressive and tensile strengths of concrete and are recomputed with variations in these uniaxial properties, e.g. due to initial specimen composition, or due to damage during loading etc.



(a) Triaxial Test Data



(b) Biaxial Test Data

Figure 3.2: Comparison of Ottosen Strength Envelope with (a) Triaxial and (b) Biaxial Test Data, Ottosen (1977).

Modes of failure

In addition to having a suitable failure criterion it is necessary to identify the mode of failure - cracking or crushing. This is especially important in the tension-compression regions, where cracking can precede crushing. In this study the mixed stress region is divided into a cracking zone and a crushing zone on the basis of the ratio of the maximum to minimum principal stresses as :

$$\textit{cracking}: \qquad \infty \leq \frac{\sigma_1}{\sigma_3} \leq -0.7333\frac{f_t'}{f_c'} \tag{3.4}$$

$$\textit{crushing}: \qquad -0.7333\frac{f_t'}{f_c'} \leq \frac{\sigma_1}{\sigma_3} \leq 0.0 \tag{3.5}$$

The influence of the intermediate stress, σ_2, is neglected. The criterion is similar to the one adopted by Barzegar and Schnobrich (1986) for two-dimensional applications. The principal stresses employed in equations 3.4 and 3.5, σ_1, σ_2 and σ_3 are arranged in descending order ($\sigma_1 \geq \sigma_2 \geq \sigma_3$). The behavior of concrete up to and after crushing is discussed in the next section. The post-cracking response of concrete is discussed in chapter 4.

3.2.2 Stress-Strain Response of Uncracked Concrete

In the pre-cracked phase, the stress-strain relationship for plain concrete is determined using a 3-D constitutive model proposed by Ottosen (1979) . In this isotropic nonlinear elastic model the total stresses are related to the total strain as follows:

$$\begin{pmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{xz} \end{pmatrix} = \frac{E_{co}}{(1-2\nu)(1+\nu)} \begin{pmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{pmatrix} \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{xz} \end{pmatrix} \tag{3.6}$$

where E_{co} and $\nu = \nu_{co}$ are the secant stiffness modulus and Poisson's ratio respectively for concrete. These moduli are functions of the states of stress and are modified as explained in the following sections. This model is capable of simulating the unloading and reloading response of concrete along a secant path as shown in figure 3.3.

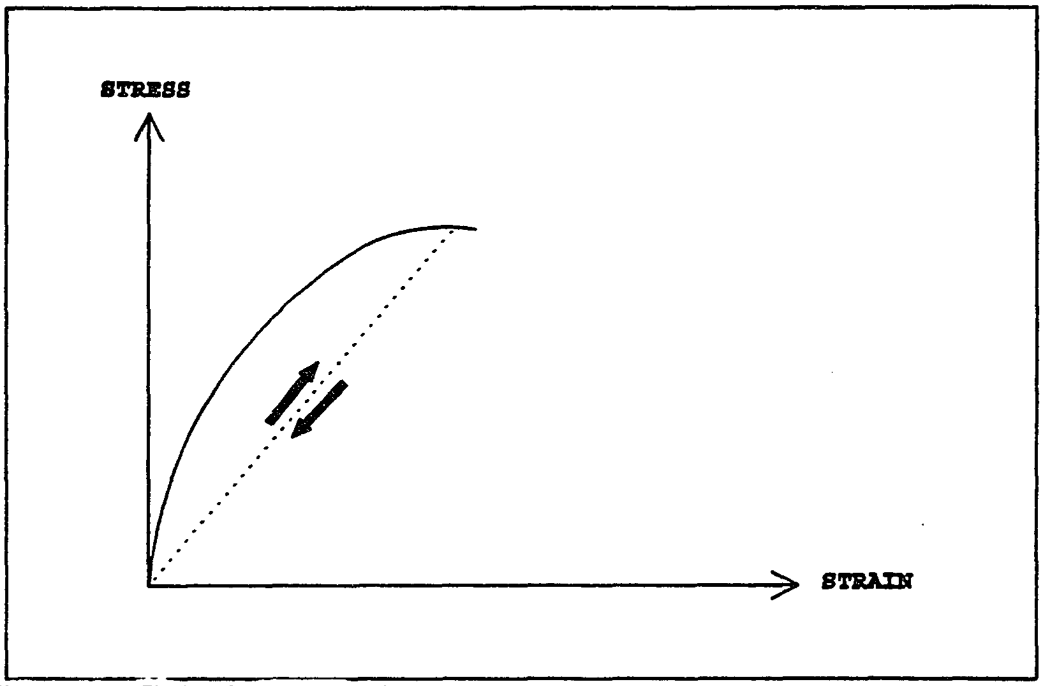


Figure 3.3: Unloading and Reloading Response of Concrete Along a Secant Path.

3.2.2.1 Computation of Pre-Peak Secant Stiffness Modulus E_{co}

The secant stiffness of concrete E_{co} employed in the model is computed (Ottosen 1979) as:

$$E_{co} = \frac{E_i}{2} - \gamma(\frac{E_i}{2} - E_f) \pm \sqrt{[\frac{E_i}{2} - \gamma(\frac{E_i}{2} - E_f)]^2 + E_f^2\gamma[D(1-\gamma) - 1]} \quad (3.7)$$

where the positive sign is taken for the ascending branch (pre-peak) and the negative sign for the descending branch (post-peak). Here E_i is the initial secant stiffness, E_f is the stiffness at triaxial compressive failure, γ is the nonlinearity index associated with the proximity of the stress-point to the failure surface and D is a post-peak softening parameter.

The nonlinearity index, γ, is a measure of the actual stresses with respect to the failure envelope (figure 3.4) and is defined as (Ottosen, 1979) :

$$\gamma = \frac{\sigma_3}{\sigma_{3f}} \tag{3.8}$$

Here σ_3 is the actual minimum principal stress (compressive) and σ_{3f} is the corresponding failure stress value, provided that the other principal stresses, σ_1 and σ_2 remain unchanged ($\sigma_1 \geq \sigma_2 \geq \sigma_3$). Thus $\gamma < 1$, $\gamma = 1$ and $\gamma > 1$ correspond to stress states inside, on, and outside the failure surface, respectively.

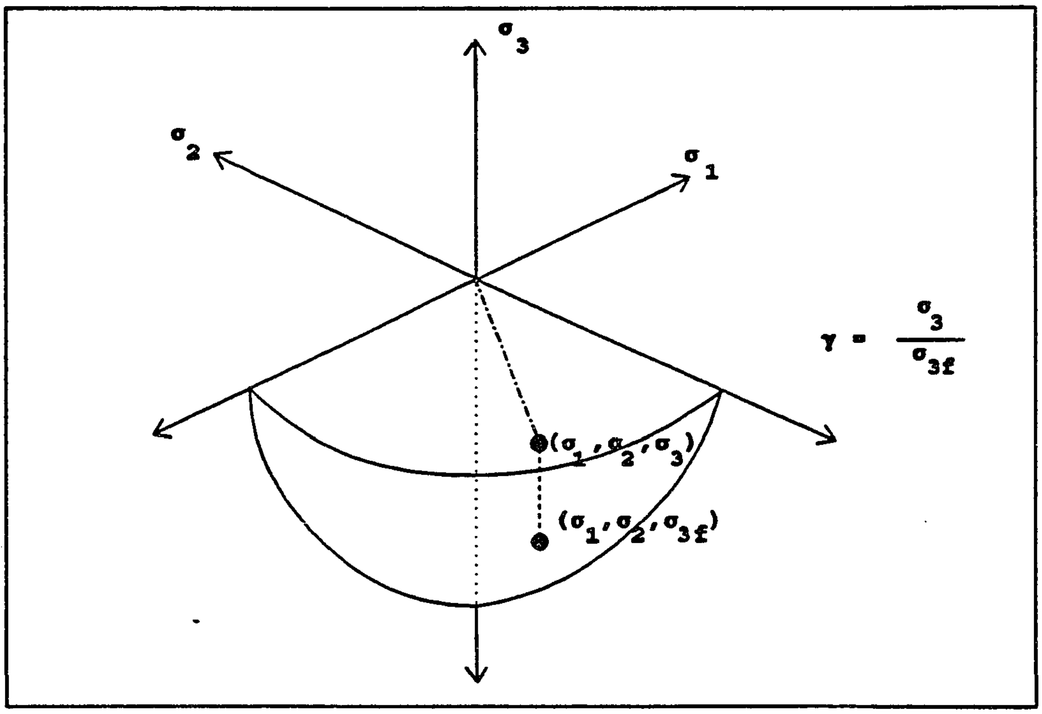


Figure 3.4: Determination of the Nonlinearity Index γ.

Ottosen (1979) has proposed a modification to this criterion when tensile stresses are present, since the concrete behavior is more linear with increased tensile stresses. When at least one principal stress, σ_1, is tensile the stress state is modified by superposing a hydrostatic pressure, $-\sigma_1$ on it, thereby producing a biaxial compressive stress state $(\sigma_1', \sigma_2', \sigma_3') = (0, \sigma_2 - \sigma_1, \sigma_3 - \sigma_1)$. The nonlinearity index

γ is:

$$\gamma = \frac{\sigma'_3}{\sigma'_{3f}} \tag{3.9}$$

where σ'_{3f} is the failure stress state provided that σ'_1 and σ'_2 are unchanged. This approach reduces the γ value reflecting linearity (equation 3.9) and $\gamma \leq 1$ always holds.

The failure stiffness E_f is a function of the loading, type of concrete etc. and is computed as (Ottosen, 1979):

$$E_f = \frac{E_c}{1 + 4(A-1)X} \tag{3.10}$$

in which $E_c = f'_c/\epsilon'_c$ is the stiffness at peak uniaxial compressive stress, and $A = E_i/E_c$. In equation 3.10, X represents the dependence on the actual loading and is expressed as:

$$X = \{\frac{\sqrt{J_2}}{f'_c}\}_f - \frac{1}{\sqrt{3}} \tag{3.11}$$

where J_2 is calculated from the same failure stress state which is used to evaluate γ, and is expressed as:

$$J_2 = \frac{1}{3}\{\sigma_1^2 + \sigma_2^2 + \sigma_{3f}^2 - \sigma_1\sigma_2 - \sigma_2\sigma_{3f} - \sigma_1\sigma_{3f}\} \tag{3.12}$$

When tensile stresses are present, $E_f = E_c$ is assumed.

The parameter D (equation 3.7) determines the post peak stress-strain curves. However, there are limitations in selecting D if the stress-strain curves should reflect the following requirements (Ottosen 1979):

- An increasing function with no inflexion points prior to failure.
- A decreasing function with at most one inflexion point after failure.
- A residual strength of zero after very large strains.

To achieve these features A $\geq 4/3$ is retained (equation 3.10). A further restriction on D, enforced by the following relations, is:

$$(1 - \frac{A}{2})^2 \leq D \leq 1 + A(A-2) \qquad A \leq 2 \tag{3.13}$$

$$0 \leq D \leq 1.0 \qquad A \geq 2 \tag{3.14}$$

3.2.2.2 Post-Peak Stiffness Computation

The post-peak behavior where some small stresses remain is modeled by the following procedure proposed by Ottosen (1979). At peak (point A in figure 3.5) the stresses result in a nonlinearity index $\gamma_f \leq 1$. In figure 3.5 the curve AB is obtained by shifting the descending curve MN parallel to the horizontal axis. The secant stiffness in the post-peak region is expressed as:

$$E_{co} = \frac{\gamma E_{MN} E_A E_M}{\gamma E_A E_M + \gamma_f E_{MN}(E_M - E_A)} \tag{3.15}$$

E_{MN} is obtained using the actual γ (current stress state) along with the negative sign in equation 3.7. E_A and E_M are the ascending (pre-peak) and descending (post-peak) stiffnesses computed using $\gamma = \gamma_f$, the value of γ at ultimate, along with a positive and negative sign, respectively, in equation 3.7. Thus, there is a gradual change in the post-peak behavior with changing stress states.

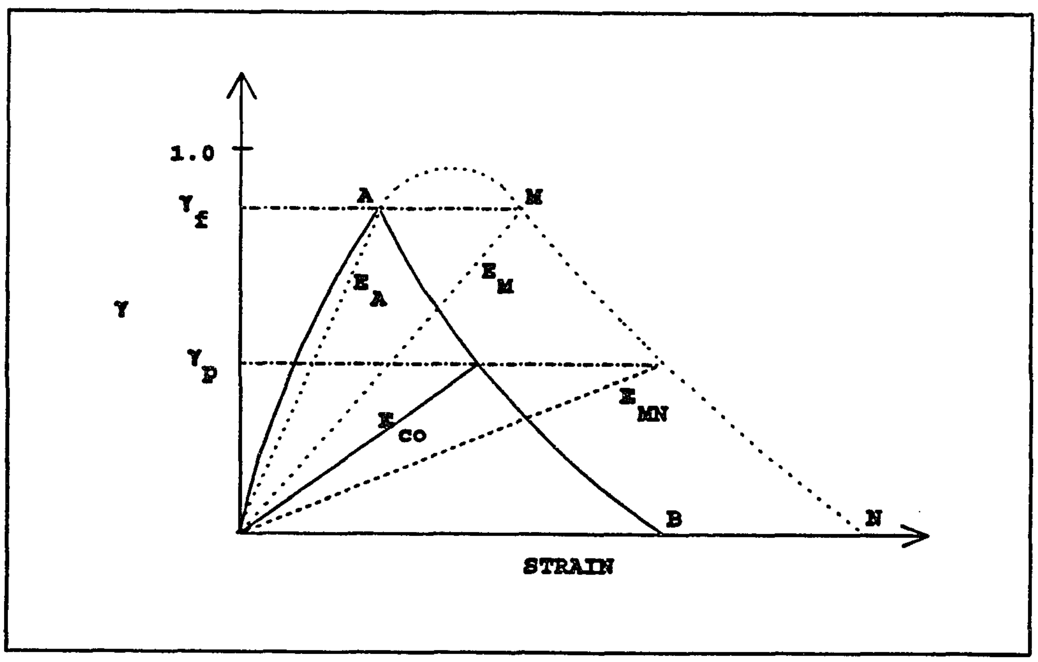


Figure 3.5: Computation of Secant Stiffness, E_{co} in the Post-Peak Region.

3.2.2.3 Changes in Poisson's Ratio ν_{co}

For uniaxial, biaxial, and triaxial compression in concrete the volumetric change is initially that of compaction followed by dilatation. Thus, the Poisson's ratio has been generalized for multiaxial states of stress by Ottosen (1979) in terms of the nonlinearity index γ as (figure 3.6):

$$\nu_{co} = \nu_i \qquad \gamma \leq \gamma_a \tag{3.16}$$

$$\nu_{co} = \nu_f - (\nu_f - \nu_i)\sqrt{1 - [\frac{\gamma - \gamma_a}{1 - \gamma_a}]^2} \qquad \gamma \geq \gamma_a \tag{3.17}$$

ν_i and ν_f are the initial secant Poisson's ratio and its value at peak compressive stress respectively. The parameter, γ_a has a value of 0.80. The second expression is valid only up to failure but it is assumed that dilatation continues beyond failure. In the post-peak zone, ν_{co} is modified along with E_{co}. However, this change is related to the bulk modulus at failure, K_f, which is kept constant beyond the peak stress. For a given post-peak stiffness E_s a Poisson's ratio ν^* is obtained with the bulk modulus, K_f, remaining unchanged. The secant Poisson's ratio is then obtained as (Ottosen, 1979):

$$\nu_{co} = 1.005\nu^* \tag{3.18}$$

The Poisson's ratio $\nu_{co} < 0.5$ is always maintained. The value of the parameter ν_f (Ottosen, 1979) is:

$$\nu_f = 0.36 \tag{3.19}$$

3.2.2.4 Input Parameters

The input parameters required for this material model are the initial secant stiffness E_i, the uniaxial compressive strength f'_c, ϵ_c the strain at peak uniaxial compressive stress, the initial Poisson's ratio ν_i, and the post-peak softening parameter D. These quantities are all user supplied values based on appropriate test data.

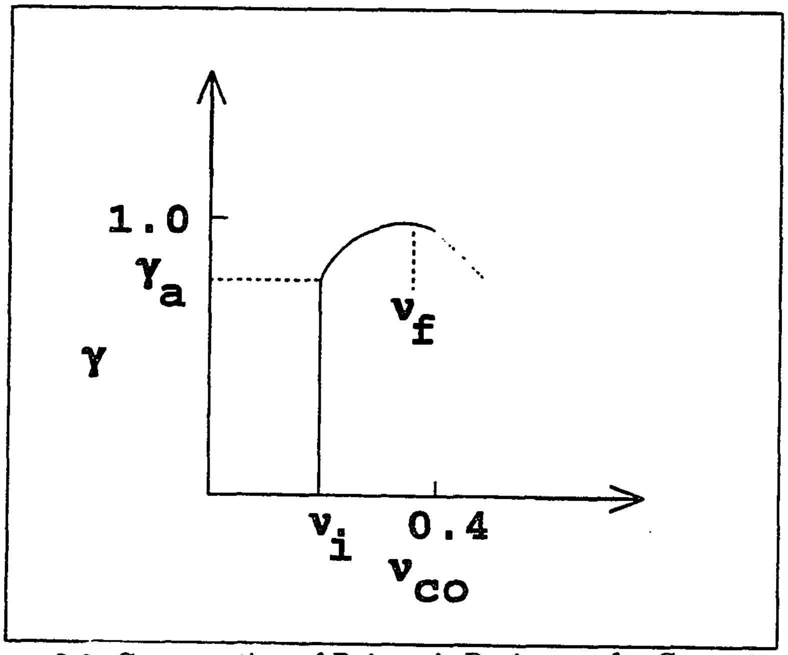


Figure 3.6: Computation of Poisson's Ratio, ν_{co}, for Concrete.

3.2.3 Correlations with Test Results

The failure envelope and stress-strain relationships discussed in the preceeding sections have been implemented in a special purpose finite element program. The details of the implementation are discussed in chapter 5. The implementation of the concrete model has been ratified by comparing the computed results obtained in the present study with the test results (Linse and Aschl 1976, Schickert and Winkler 1977) and also analytical results obtained by Ottosen (1981, 1982). The analytical results obtained by Ottosen (1981, 1982) are the only ones included for comparisons because they provide a means of ratification for its implementation in the present study.

In figure 3.7 stress-strain response from a uniaxial compression test along with the results obtained in the present study are shown; results obtained by Ottosen (1981,1982) have also been included. The uniaxial material properties employed in the various analyses are presented on the figures. Figures 3.8 and 3.9 shows the comparisons of triaxial stress-strain response along the compression meridian (loading path as indicated on the figures) with the test results; results obtained by Ottosen (1981,1982) are similar to the results obtained in the present study. Triaxial stress-strain responses along the tensile meridian (Loading path as indicated on the figure) are shown in figure 3.10.

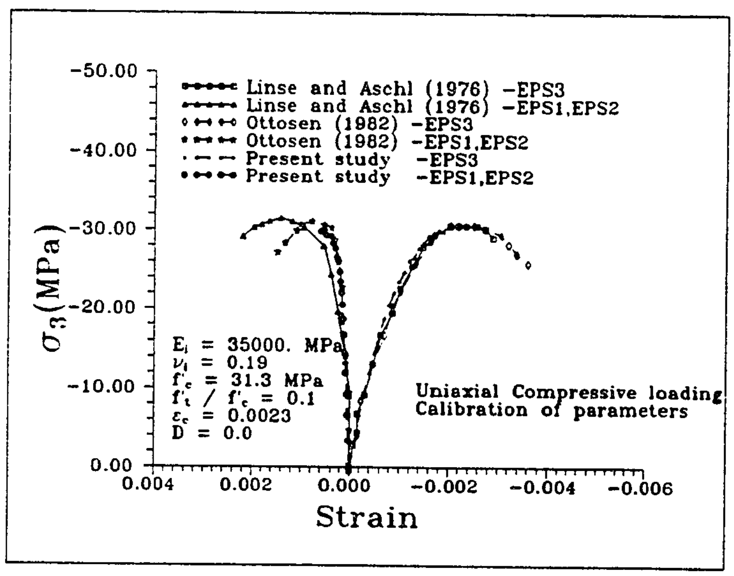


Figure 3.7: Comparison with Uniaxial Compressive Loading Test Data.

While all of the above examples represent monotonic proportional loadings, Ottosen (1981,1982) has shown that the material model performs adequately for certain classes of non-proportional loadings as well. Figure 3.11 shows results from a test where the hydrostatic loads are constant throughout the loading. However, after a certain hydrostatic stress level (-25.5 MPa), the load is applied along the compressive meridian while keeping the hydrostatic stress constant. The analytical results obtained in this study are similar to the results obtained by Ottosen

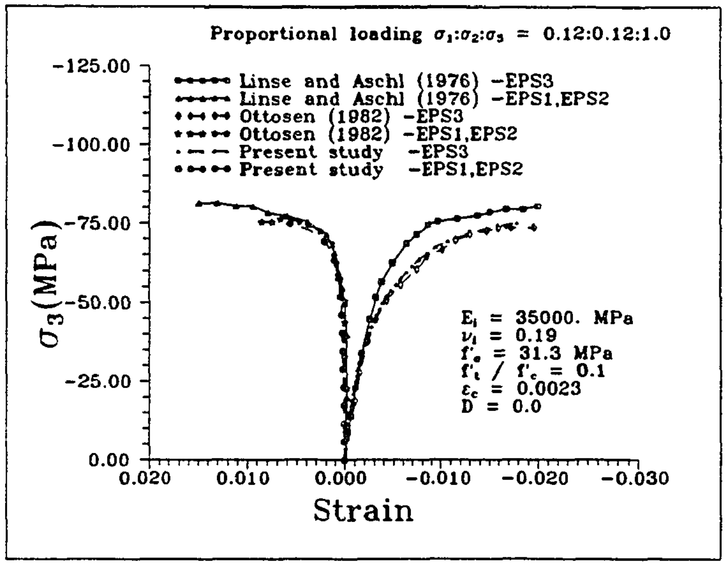


Figure 3.8: Triaxial Loading Along the Compressive Meridian.

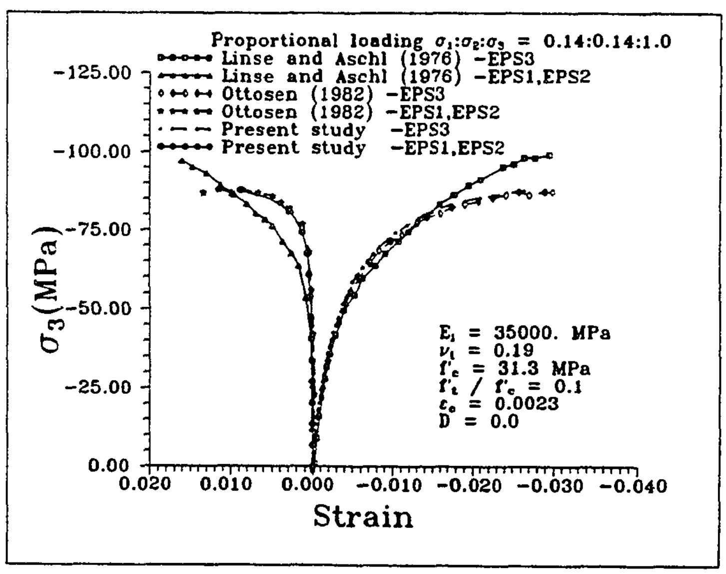


Figure 3.9: Triaxial Loading Along the Compressive Meridian.

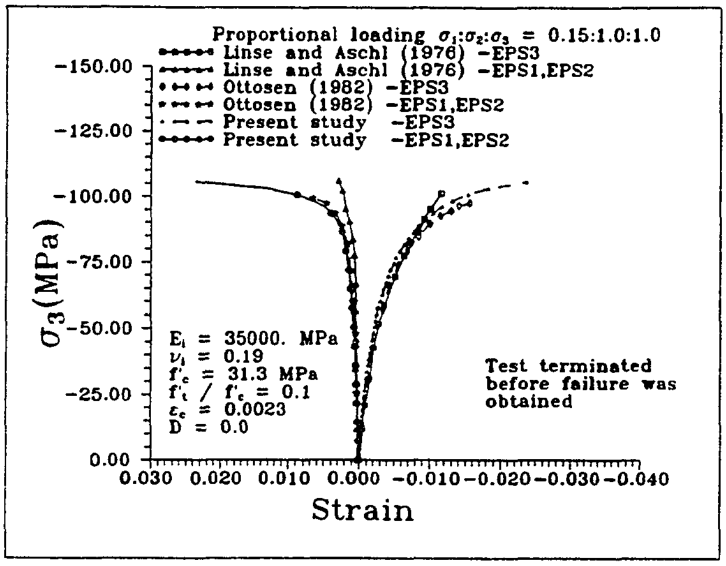


Figure 3.10: Triaxial Loading Along the Tensile Meridian.

(1981,1982) and compare well with the test results.

The analytical predictions obtained in the present study compare well with the experiments and also the predictions obtained by Ottosen (1981,1982). The dominant nonlinear response characteristics appear to be well represented by Ottosen's model employed in the present study.

3.3 Stress-Strain Response of Steel

The steel reinforcement is assumed to be smeared into equivalent membrane layers with uniaxial properties oriented in each reinforcing direction. An elasto-plastic response with possible strain-hardening, identical in tension and compression, is assumed (figure 3.12). The constitutive matrix in an orthogonal coordinate system (x', y', z') with the bar direction coinciding with the x' is:

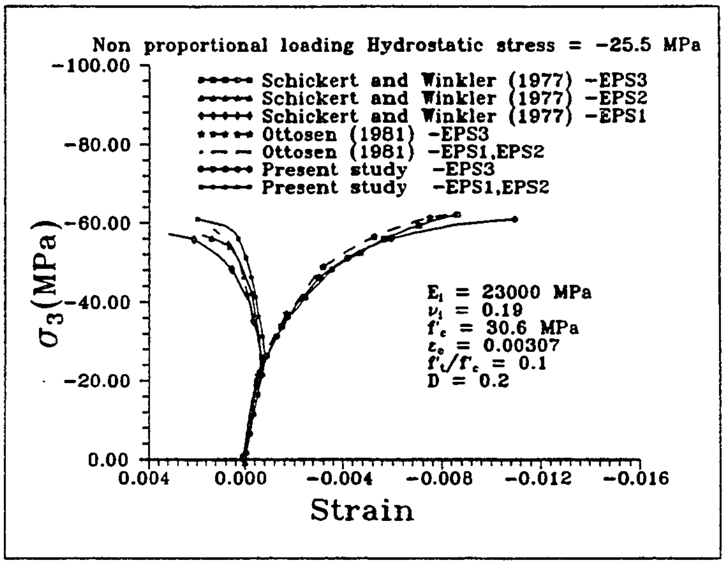


Figure 3.11: Comparison with Triaxial Non-proportional Loading Data.

$$[D] = \begin{pmatrix} E_s & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.20}$$

The modulus of elasticity E_s is modified based on the following expressions:

$$E_s = E_y \qquad |\epsilon_s| < |\epsilon_y| \tag{3.21}$$

$$E_s = \alpha E_y + |\frac{\sigma_y}{\epsilon_s}|(1-\alpha) \qquad |\epsilon_s| \geq |\epsilon_y| \tag{3.22}$$

Where σ_y is the yield stress for steel, E_y the initial secant stiffness, and α is a strain-hardening parameter expressed as the ratio of hardening stiffness to the initial secant stiffness.

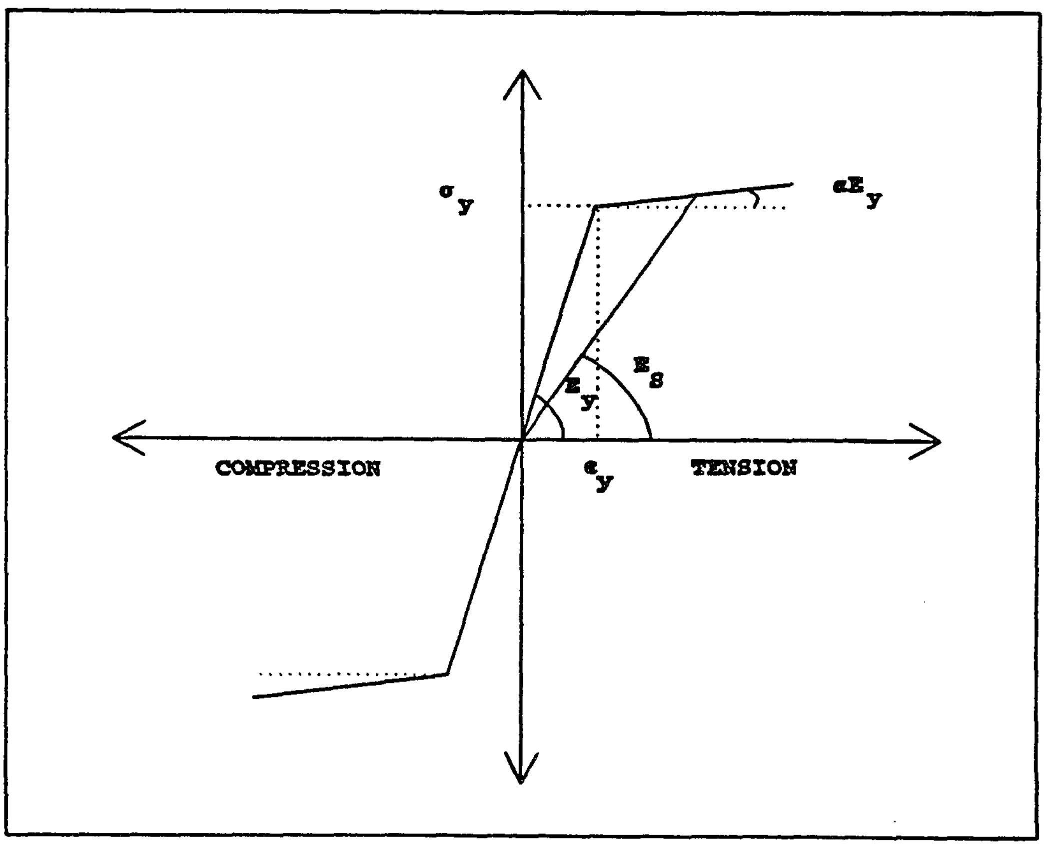


Figure 3.12: Stress-Strain Response of Steel

Chapter 4

Post-Cracking Response of Plain and Reinforced Concrete

4.1 Introduction

The weakness of concrete in carrying tensile stresses and the resulting formation of cracks is a major factor in the nonlinear response of reinforced concrete structures. After cracking, strain softening causes the strength and stiffness of plain concrete to gradually reduce in the direction normal to the cracking plane. The load carrying capacity and the stress distribution are significantly altered due to this effect.

After cracking has occured in reinforced concrete, the tension-stiffening in the vicinity of the reinforcement enhances the stiffness and stress carrying capacity of concrete near the tension region. The shear stress transferred along the crack interfaces is controlled by the aggregate interlock mechanism. The crack opening during the sliding action, referred to as shear dilatancy, is resisted by the reinforcement crossing the crack. Further, the sliding action along the cracked surface is directly resisted by the reinforcement through the dowel action. Due to cracking and the release of confining effect in the crack normal direction, concrete compressive strength and stiffness parallel to the crack direction diminishes.

In this chapter the post-cracking behavior of plain and reinforced concrete is described and material models to simulate these phenomena are proposed. To facilitate this development, rheological models representing the various phenomena are described.

4.2 Plain Concrete Under Uniaxial Tension

A plain concrete element subjected to a general state of stress, at incipient cracking is shown in figure 4.1. In formulating the post cracking concrete behavior, an approach similar to the one used by deBorst and Nauta (1985) is adopted. In this approach the total strain ϵ_n^t is resolved into intact concrete strains ϵ_n^{co} and crack strains ϵ_n^{cr}, i.e. $\epsilon_n^t = \epsilon_n^{co} + \epsilon_n^{cr}$. The constitutive matrix for the intact concrete $[D^{co}]$ is given in equation 3.6 while that for the crack (deBorst and Nauta, 1985) is:

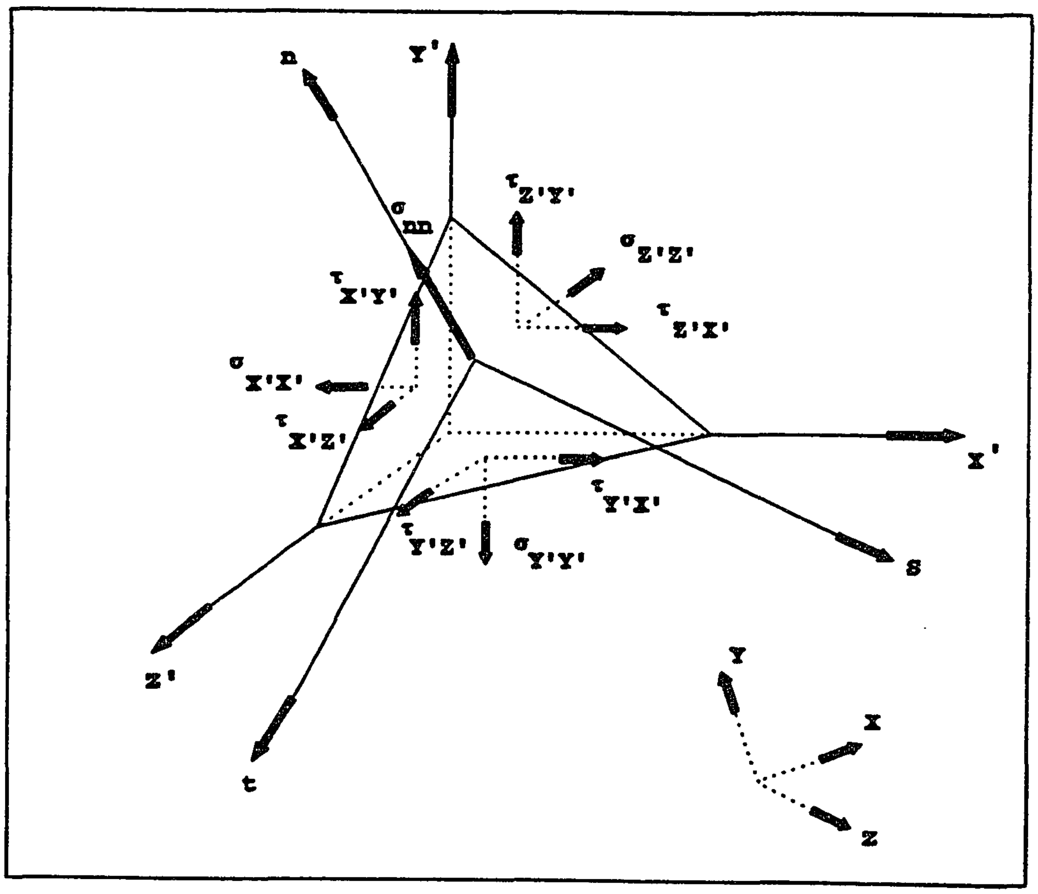


Figure 4.1: Plain Concrete Element at Incipient Cracking

$$[D]^{cr} = \begin{pmatrix} E^{cr}_{nn} & 0 & 0 \\ 0 & G^{cr}_{ns} & 0 \\ 0 & 0 & G^{cr}_{nt} \end{pmatrix} \tag{4.1}$$

where E^{cr}_{nn} is the crack stiffness normal to the crack plane (n) and G^{cr}_{ns} and G^{cr}_{nt} are the crack interface shear stiffness components along the crack plane (s-t). The crack normal stiffness is determined from the bilinear stress-strain curve shown in figure 4.2(e), where G_f is the fracture energy, assumed to be a material constant. The formulation for the crack interface shear stiffness is discussed in section 4.3.4.

The post-cracking constitutive matrix of plain concrete in the global (x,y,z) coordinate system due to strain softening is formulated as (de Borst and Nauta, 1985):

$$[D]^{cr-con}_{x,y,z} = [D]^{co} - [D]^{co}[N]\{[D]^{cr} + [N]^T[D]^{co}[N]\}^{-1}[N]^T[D]^{co} \tag{4.2}$$

where [N] represents the transformation of the crack local strains to the global directions given by:

$$(N) = \begin{pmatrix} l_x^2 & l_x l_y & l_z l_x \\ m_x^2 & m_x m_y & m_z m_x \\ n_x^2 & n_x n_y & n_z n_x \\ 2l_x m_x & l_x m_y + l_y m_x & l_z m_x + l_x m_z \\ 2m_x n_x & m_x n_y + m_y n_x & m_z n_x + m_x n_z \\ 2n_x l_x & n_x l_y + n_y l_x & n_z l_x + n_x l_z \end{pmatrix} \tag{4.3}$$

where l_x, m_x and n_x are the direction cosines of the normal to the crack plane (n) with respect to the global directions (x,y,z). Similarly l_y, m_y and n_y and l_z, m_z and n_z represent the direction cosines of two orthogonal directions (s-t), on the crack plane.

For the plain concrete tensile specimen shown in figure 4.2 (a), using equation 4.2, the post-cracking stiffness matrix can be shown to be (assuming the crack

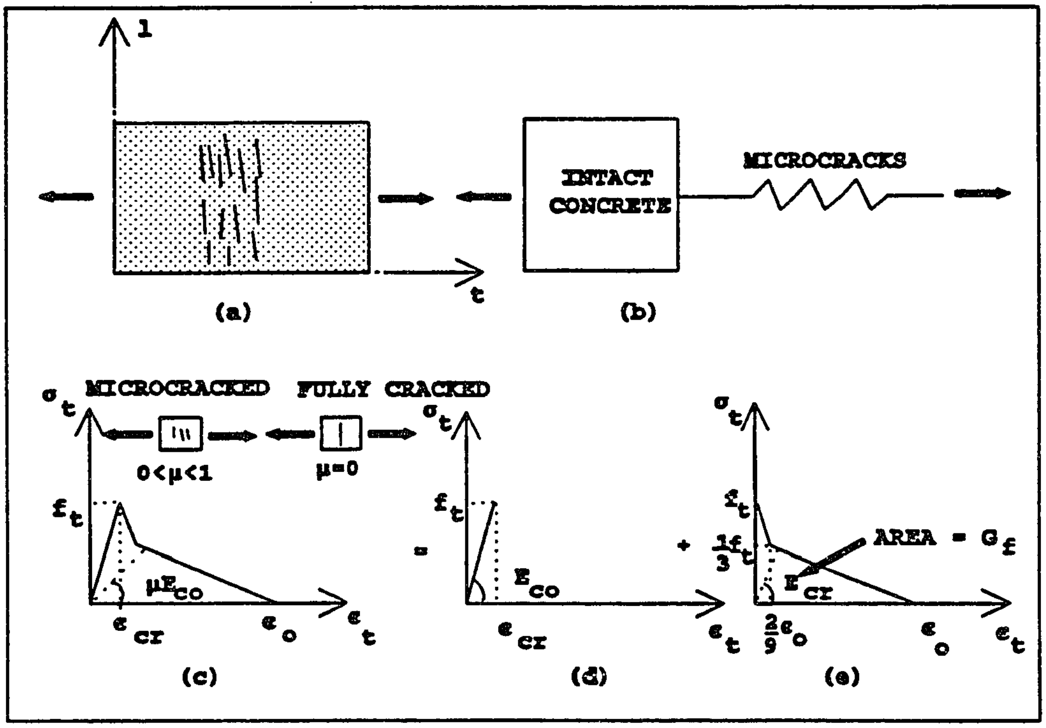


Figure 4.2: (a) Cracked Plain Concrete Specimen, (b) Rheological Representation of Plain Concrete Cracking, (c) Stress-Strain Relationship For Cracked Plain Concrete, (d) Stress-Strain Relationship for Uncracked Concrete, (e) Stress-Strain Relationship for Cracks.

normal is along the global x direction):

$$[D]^{cr-con}_{x,y,z} = E'' \begin{pmatrix} \frac{(1-\nu)\mu A}{B} & \frac{\nu\mu A}{B} & \frac{\nu\mu A}{B} & 0 & 0 & 0 \\ \frac{\nu\mu A}{B} & \frac{D}{B} & \frac{\nu F}{B} & 0 & 0 & 0 \\ \frac{\nu\mu A}{B} & \frac{\nu F}{B} & \frac{D}{B} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\beta(1-2\nu)}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{(1-2\nu)}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\beta(1-2\nu)}{2} \end{pmatrix} \tag{4.4}$$

where:

$$\begin{aligned} E'' &= \frac{E_{co}}{(1-2\nu)(1+\nu)} \\ A &= 1-\nu-2\nu^2 \\ B &= 1-\nu-2\mu\nu^2 \\ D &= (1-2\nu)(1-\mu\nu^2) \\ F &= (1-2\nu)(1+\mu\nu) \end{aligned} \tag{4.5}$$

The parameter μ represents the degree of damage in the secant stiffness of concrete and β represents the amount of damage in the shear stiffness. When the crack is fully open (μ=0.0), the stiffness normal to the crack and the coupling stiffness relating the cracked and uncracked directions disappear. A rheological representation of the plain concrete tensile specimen is shown in figure 4.2 (b). Figure 4.2 (c), (d), (e) show the stress-strain relationships of the composite cracked concrete, the intact concrete and the crack, respectively.

4.3 Modelling of Cracked Reinforced Concrete

4.3.1 RC Bar Subjected to Uniaxial Tension

In establishing the stiffness of a simple RC bar element figure 4.3(a), tension-stiffening represents the ability of the intact concrete between adjacent cracks to transmit tensile stresses. In this case, the post-cracking response is dominated by the presence of the reinforcement. One possible rheological model is depicted in figure 4.3 (b). Here, the fracture energy G_f of plain concrete has been artificially increased by ΔG_f. This increase in energy dissipation capacity represents an average strain energy associated with the concrete-reinforcement interface bond stresses which cause distributed cracking. The value of ΔG_f is usually much higher than G_f, which implies that the tension stiffening effect dominates the energy dissipation characteristics of the cracked reinforced concrete element.

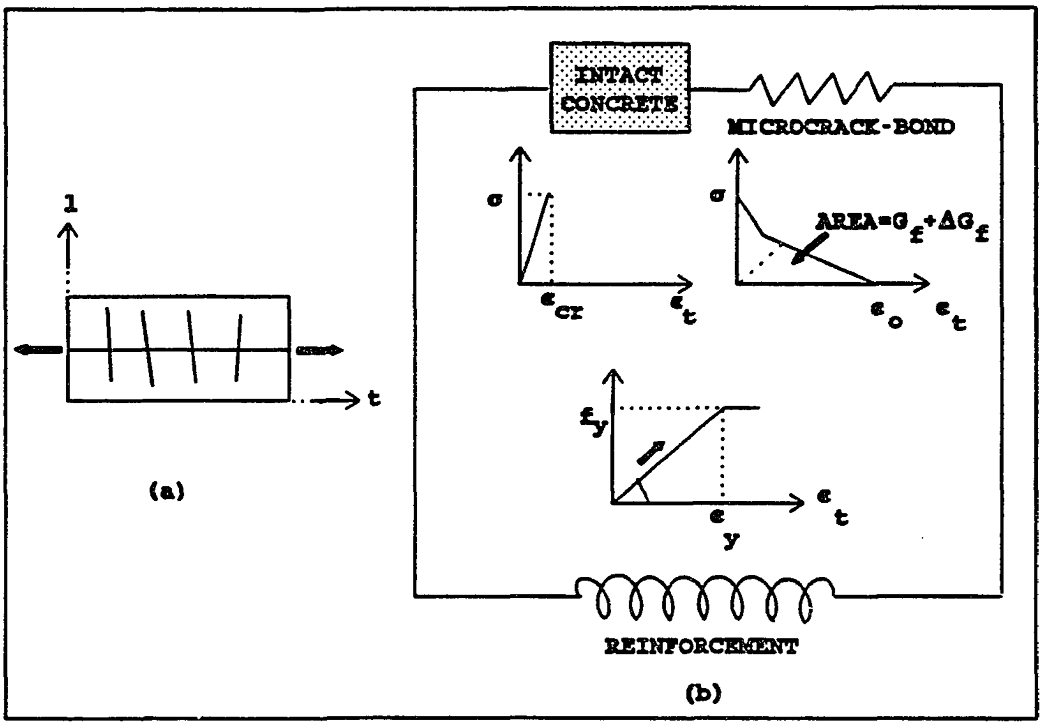


Figure 4.3: (a) RC Element, (b) Rheological Representation of Reinforced Concrete Cracking, Rots (1988)

For the RC tensile specimen the curve representing the combined effects of tension softening and tension stiffening may be constructed based on measured post cracking responses of RC test specimen under uniaxial tension. But, in cases where the cracks are non-orthogonal to the reinforcing direction, or where crack directions shift and additional crack formation is possible, this procedure meets with difficulty. This is because the strain softening behavior is dominant in the crack normal direction, while the tension-stiffening is effective in the reinforcement direction(s).

To rectify this shortcoming, an alternate rheological model is proposed as shown in figure 4.4. It is assumed that on cracking, the normal stiffness in the microcrack-intact concrete assembly is completely lost - $E_{co}/1000$ is used for numerical stability purposes. The tension-stiffening spring stiffness E_t^{ts} which now models the combined stiffness of both the bond stresses and the strain-softening effects is activated when the strain in the reinforcing direction exceeds the uniaxial cracking strain ϵ_{cr}. The constitutive matrix for the cracked concrete specimen $[D]_{x,y,z}^{cr-rc}$ including tension stiffening is then:

$$[D]_{x,y,z}^{cr-rc} = [D]_{x,y,z}^{cr-con} + [T]^T [D]^{ts} [T] \tag{4.6}$$

where:

$$[D]^{ts} = \begin{pmatrix} E_t^{ts} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{4.7}$$

and [T] transforms $[D]^{ts}$ from the steel direction (x', y', z') to the global direction (x, y, z) given by:

$$[T] = \begin{pmatrix} l_x^2 & m_x^2 & n_x^2 & l_x m_x & m_x n_x & n_x l_x \\ l_y^2 & m_y^2 & n_y^2 & l_y m_y & m_y n_y & n_y l_y \\ l_z^2 & m_z^2 & n_z^2 & l_z m_z & m_z n_z & n_z l_z \\ 2l_x l_y & 2m_x m_y & 2n_x n_y & l_x m_y + l_y m_x & m_x n_y + m_y n_x & n_x l_y + n_y l_x \\ 2l_z l_y & 2m_z m_y & 2n_z n_y & l_z m_y + l_y m_z & m_z n_y + m_y n_z & n_z l_y + n_y l_z \\ 2l_x l_z & 2m_x m_z & 2n_x n_z & l_x m_z + l_z m_x & m_x n_z + m_z n_x & n_x l_z + n_z l_x \end{pmatrix}$$

l,m,n are direction cosines relating the tension stiffening directions (same as the steel directions) to the global directions.

With this procedure the memory of cracking is retained and possible changes in the shear interface stiffness may be taken into account. Additionally, due to establishment of the tension stiffening spring in the direction of the reinforcement, the

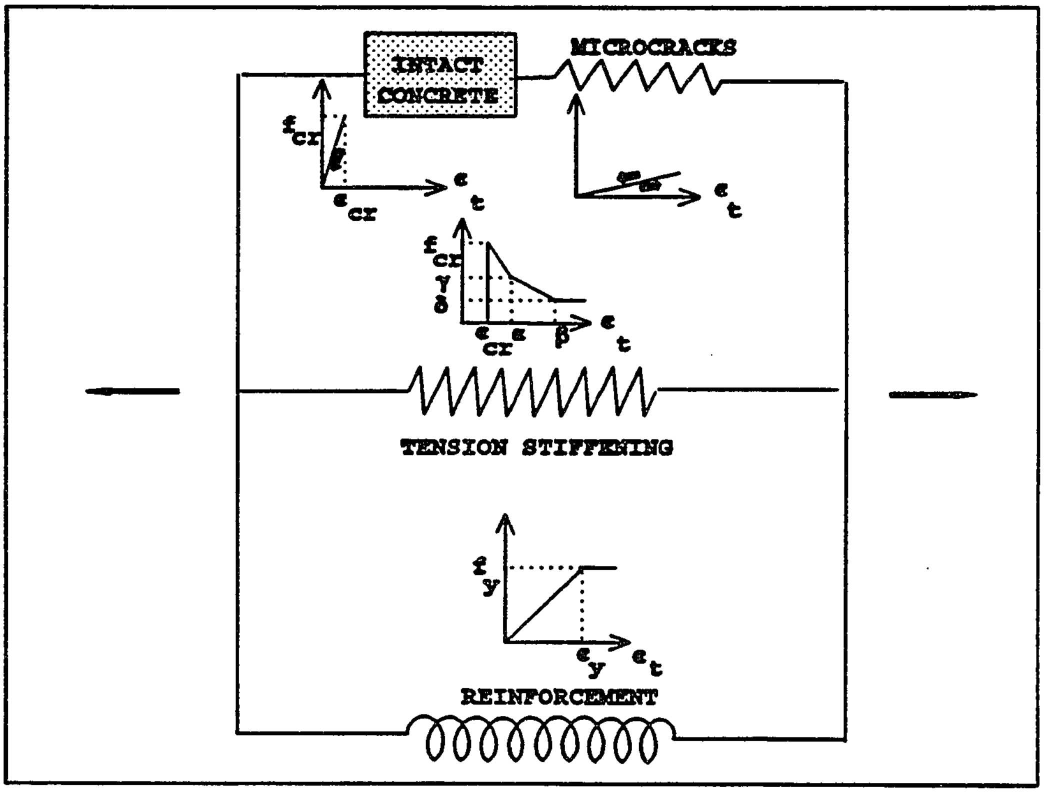


Figure 4.4: Rheological Representation of Reinforced Concrete Cracking, Present Study

stiffness contribution of the bond stresses are modeled in their proper orientation. The shape of the tension-stiffening spring (figure 4.4) is determined by analyzing the results of number of a number of tests on RC specimen. The curve is expressed in terms of known quantities vis. cracking strength of concrete and yield strain of the reinforcement. The values of $\alpha, \beta, \gamma, \delta$ are established through comparison of the tensile stress-strain response with uniaxial tensile test results (presented in section 6.2.1).

It should be noted that when the crack normal stiffness component is completely lost in the manner described above, the cracked reinforced concrete assembly does not retain any Poisson's effect. This is easily seen form equation 4.4. However, experimental evidence presented by Kolleger (1988) and Dyngland (1989) indicate that after cracking some cross coupling stiffness is still present; they observed that for a previously cracked RC element, compressive loading in the uncracked orthogonal direction caused significant straining in the cracked direction. This coupling disappears gradually with increasing damage in either direction due to loading. In the present study this coupling effect is included and is expressed in terms of the tension-stiffening stiffness components in the direction of the reinforcement and is expressed as:

$$[D]^{Coupling}_{x',y',z'} = \begin{pmatrix} 0 & \nu\sqrt{E_1E_2} & \nu\sqrt{E_1E_3} & 0 & 0 & 0 \\ \nu\sqrt{E_1E_2} & 0 & \nu\sqrt{E_2E_3} & 0 & 0 & 0 \\ \nu\sqrt{E_1E_3} & \nu\sqrt{E_2E_3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{4.8}$$

where E_1, E_2 and E_3 are normal stiffness components in three mutually orthogonal directions including atleast one reinforcement direction. This coupling matrix is transformed to the global (x,y,z) directions and then added to the cracked concrete stiffness matrix expressed by equation 4.6 to result in the stiffness matrix of the RC bar as:

$$[D]^{cr-rcon}_{x,y,z} = [D]^{cr-con}_{x,y,z} + [T]^T[D]^{ts}[T] + [T]^T[D]^{Coupling}[T] \tag{4.9}$$

For example, for the uniaxially reinforced bar shown in figure 4.4 (a) these terms are $E_1 = E_t^{ts}$, where E_t^{ts} is the secant stiffness of the tension stiffening spring along the steel direction (equation 4.7) and $E_2 = E_3 = E^{co}$, Where E^{co} is the stiffness of the concrete in the remaining two unreinforced directions provided that the strains in these two directions are compressive. Tensile strains in an unreinforced direction would imply an absence of coupling as in the plain concrete response. According to equation 4.8 the coupling stiffness terms are present until either tension-stiffening effect is exhausted completely or the concrete crushes.

4.3.2 General Triaxial Loading of RC

In the preceeding section the behavior of a uniaxially reinforced concrete bar with the crack orthogonal to the bar direction was discussed. However, in most practical situations the orientation of the cracks are at some skew angle with the reinforcement. In such situations, it is necessary to compute the tension-stiffening spring parameters while accounting for the this skew angle . Further, it is possible for a cracked sampling point to contain steel in three orthogonal orientations and it is then necessary to compute the tension-stiffening contribution in each steel direction accordingly. Additionally, the coupling stiffness expressed by equation 4.8 must also account for these tension-stiffening contributions in their proper directions. In this section RC elements with three different cases of orthogonal arrangements of reinforcement are considered.

4.3.2.1 Tri-directional Orthogonally Reinforced Concrete Elements

Consider a hypothetical RC element subjected to a general state of loading (as shown in figure 4.5). For the purpose of this discussion the element has been considered to be anisotropically reinforced in three orthogonal directions (x', y', z'). For the given load, the equilibrium equations in the three reinforcement directions (x', y', z') at incipient cracking are:

$$\sigma_{x'} A l_1' + \rho_{x'} \sigma_{sx'} A l_1' + \tau_{x'y'} A m_1' + \tau_{x'z'} A n_1' = \sigma_{nn} A l_1' \tag{4.10}$$

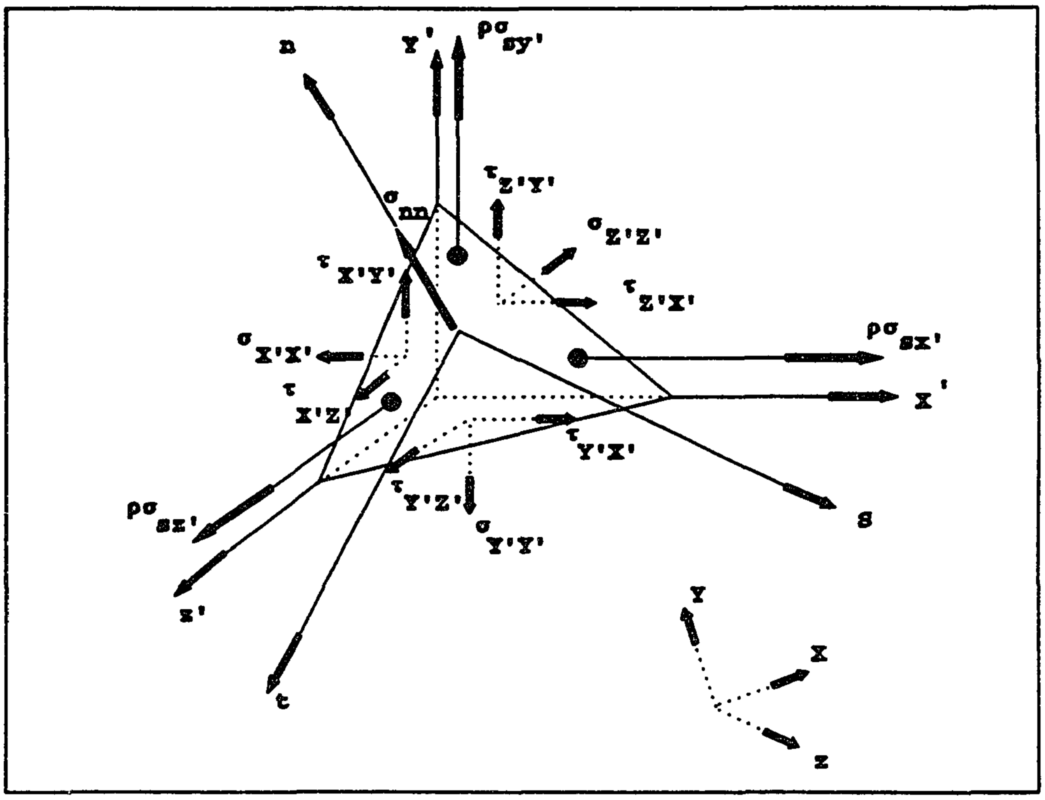


Figure 4.5: Tri-Directionally Reinforced Concrete Element Subjected to General Loads Prior to Cracking

$$\sigma_{y'} A m_1' + \rho_{y'} \sigma_{sy'} A m_1' + \tau_{x'y'} A l_1' + \tau_{y'z'} A n_1' = \sigma_{nn} A m_1' \tag{4.11}$$

$$\sigma_{z'} A n_1' + \rho_{z'} \sigma_{sz'} A n_1' + \tau_{y'z'} A m_1' + \tau_{x'z'} A l_1' = \sigma_{nn} A n_1' \tag{4.12}$$

where (l_1', m_1', n_1') , (l_2', m_2', n_2') and (l_3', m_3', n_3') are the direction cosines of the three principal stress directions with respect to the three reinforcement directions. A represents the area of the element along the crack plane. The rheological representation of this element at this stage and its schematic representation are shown in figure 4.6 (a) and (b), respectively.

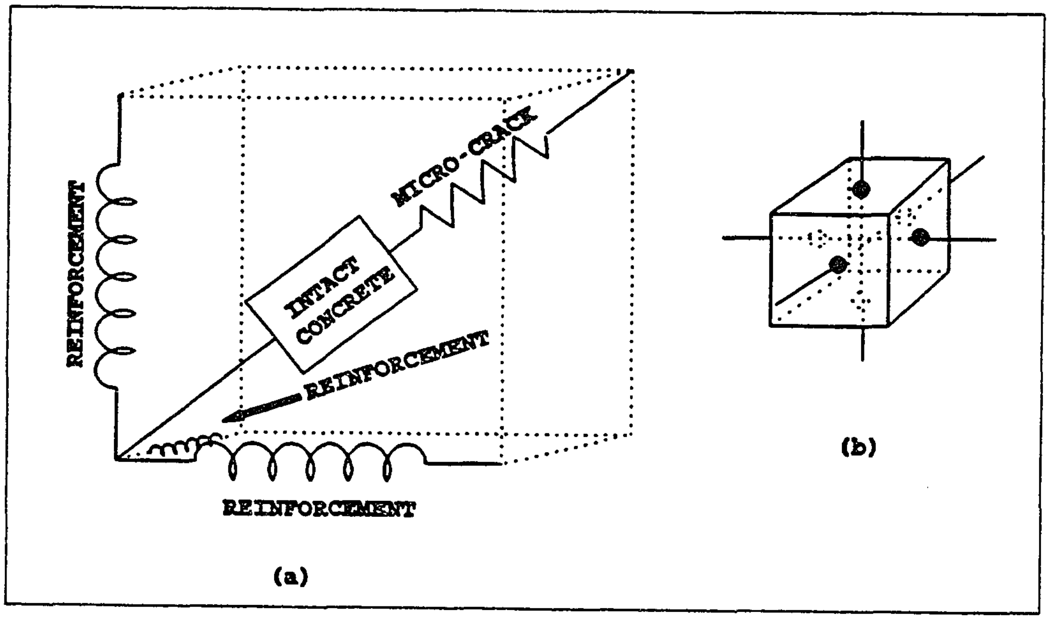


Figure 4.6: (a) Rheological Representation of a Tri-Directionally Reinforced Concrete Element Prior to Cracking (b) Schematic Representation of the Element.

When a small increment of load is applied to the specimen and it cracks, it is initially assumed that the shear stresses developing on the crack plane are negligible (the presence of reinforcement in each direction ensures a gradual build-up of the shear stress). Figure 4.7 shows the state of stress in the RC element under these conditions. The equilibrium equations in the three reinforcing directions are :

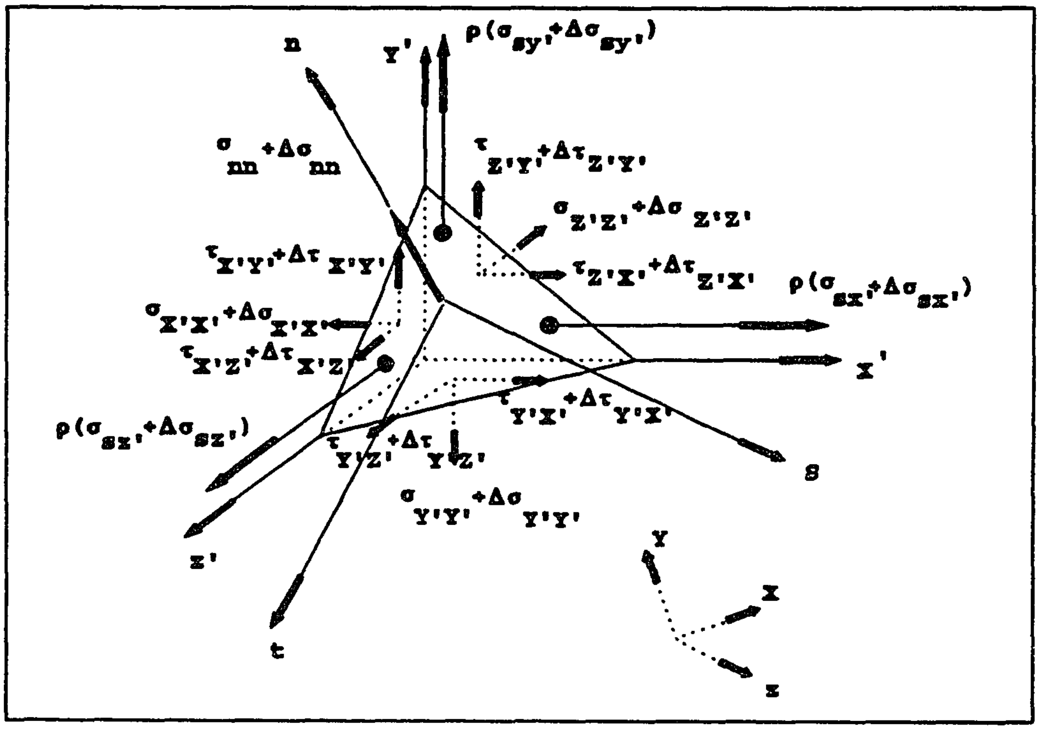


Figure 4.7: Tri-Directionally Reinforced Concrete Element Subjected to General Loads After Cracking, With Strain $\epsilon < \epsilon_{cr}$ in the Steel Direction.

$$(\sigma_{x'} + \Delta\sigma_{x'})Al_1' + \rho_{x'}(\sigma_{sx'} + \Delta\sigma_{sx'})Al_1' + (\tau_{x'y'} + \Delta\tau_{x'y'})Am_1' +$$
$$(\tau_{x'z'} + \Delta\tau_{x'z'})An_1' = (\sigma_{nn} + \Delta\sigma_{nn})Al_1' \tag{4.13}$$

$$(\sigma_{y'} + \Delta\sigma_{y'})Am_1' + \rho_{y'}(\sigma_{sy'} + \Delta\sigma_{sy'})Am_1' + (\tau_{x'y'} + \Delta\tau_{x'y'})Al_1' +$$
$$(\tau_{y'z'} + \Delta\tau_{y'z'})An_1' = (\sigma_{nn} + \Delta\sigma_{nn})Am_1' \tag{4.14}$$

$$(\sigma_{z'} + \Delta\sigma_{z'})An_1' + \rho_{z'}(\sigma_{sz'} + \Delta\sigma_{sz'})An_1' + (\tau_{y'z'} + \Delta\tau_{y'z'})Am_1' +$$
$$(\tau_{x'z'} + \Delta\tau_{x'z'})Al_1' = (\sigma_{nn} + \Delta\sigma_{nn})An_1' \tag{4.15}$$

A gradual strain softening behavior in the crack normal direction is simulated by a strain-softening spring in that direction. The assumed shape of the stress-strain response of this spring is shown in figure 4.8. The strains in each steel direction is monitored until atleast one of these reaches the uniaxial cracking strain level. At this stage the strain softening spring stiffness in the crack normal direction is released completely and the 'tension-stiffening' spring in each steel direction is activated. If the strain in a particular steel direction is below the cracking strains, tension-stiffening is still considered to be activated in that direction if these strains are tensile. If the crack orientation with respect to the reinforcement is such that the strain in a steel direction is compressive no tension stiffening spring is activated in the corresponding steel direction. Figure 4.9 shows the state of stress in the RC element after the tension-stiffening springs are activated. The incremental equilibrium equations at this state of stress is expressed by the following equations:

$$\begin{aligned}\Delta\sigma_{x'}Al_1' + \rho_{x'}\Delta\sigma_{sx'}Al_1' + \Delta\tau_{x'y'}Am_1' + \Delta\tau_{x'z'}An_1' &= \\ \Delta\sigma_{nn}Al_1' + \Delta\tau_{sn}Al_2' + \Delta\tau_{nt}Al_3'\end{aligned} \tag{4.16}$$

$$\begin{aligned}\Delta\sigma_{y'}Am_1' + \rho_{y'}\Delta\sigma_{sy'}Am_1' + \Delta\tau_{x'y'}Al_1' + \Delta\tau_{y'z'}An_1' &= \\ \Delta\sigma_{nn}Am_1' + \Delta\tau_{sn}Am_2' + \Delta\tau_{nt}Am_3'\end{aligned} \tag{4.17}$$

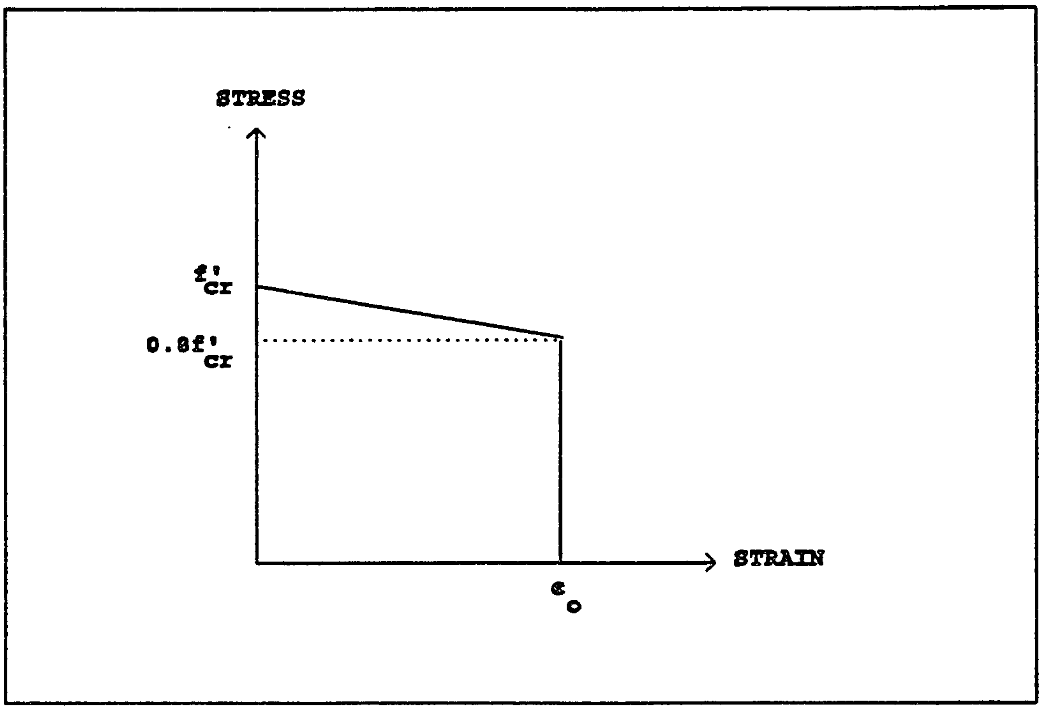


Figure 4.8: Strain-Softening Envelope Employed Normal to the Cracks in Cracked RC, With Steel Direction Strains $\epsilon < \epsilon_{cr}$.

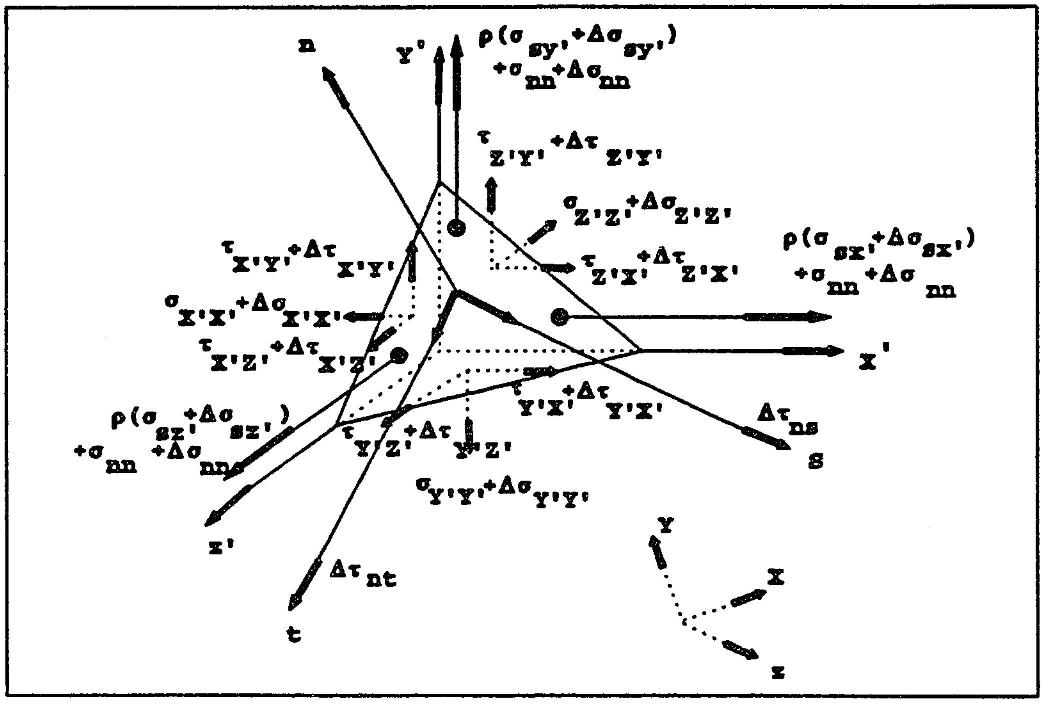


Figure 4.9: Tri-Directionally Reinforced Concrete Element Subjected to General loads, With Atleast One Strain $\epsilon \geq \epsilon_{cr}$ in the Steel Direction.

$$\begin{aligned}\Delta\sigma_{z'}An'_1 + \rho_{z'}\Delta\sigma_{sz'}An'_1 + \Delta\tau_{y'z'}Am'_1 + \Delta\tau_{x'z'}Al'_1 \;&= \\ \Delta\sigma_{nn}An'_1 + \Delta\tau_{sn}An'_2 + \Delta\tau_{nt}An'_3 & \qquad (4.18)\end{aligned}$$

It should be clear to the reader that the state of stress in the RC element shown in figure 4.7 and figure 4.9 are equivalent and that the crack normal stress can be identically replaced by an equal stress in each reinforcing spring direction. This is also evident from the above equations. Figure 4.10 (a) and (b) show the rheological representation of the element at this stage of the straining and the corresponding schematic representation of the cracked point, respectively.

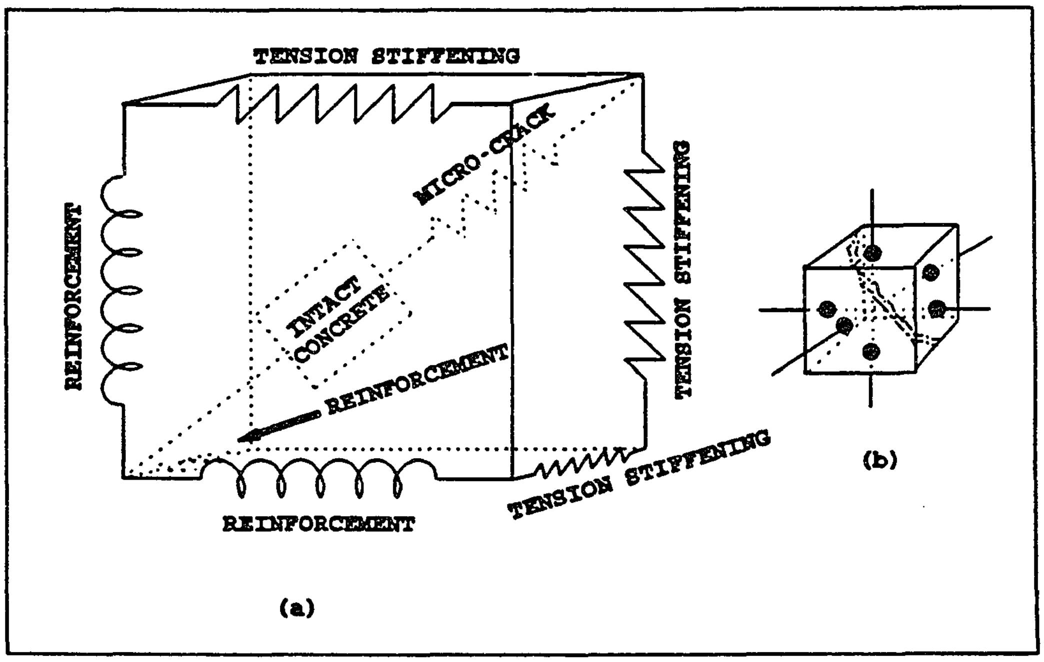


Figure 4.10: (a) Rheological Representation of a Tri-Directionally Reinforced Concrete After Full Crack Opening (b) Schematic Representation of a Cracked Point.

The following observations can be made about this approach to modelling tension stiffening:

- The tension stiffening in each reinforcing direction is thus fully defined in terms of the crack initiating stress and strain and the yield strain of the steel in the corresponding direction.
- A larger straining in a weakly reinforced direction is directly translated into rapid release of the tension-stiffening retained in that direction. However, such a directional loss of tension-stiffening (along with yielding of the steel) does not translate into a complete loss of tension-stiffening.
- The shear stresses generated on the crack plane immediately after cracking in such a case is negligible, provided neither of the reinforcements yield at cracking. If the reinforcement in one direction yields prior to the others, shear stresses do develop on the crack surface to equilibrate the unbalanced applied loads in that direction.

4.3.2.2 Bi-Directional Orthogonally Reinforced Concrete Elements

A large number reinforced concrete structural elements have planar geometry. Consequently, the reinforcement arrangement in these elements is typically a mesh reinforcement in the plane of the RC element. Consider such an element under a state of general loading (figure 4.11) . The equilibrium equations (along x', y', z') for this element at incipient cracking are:

$$\sigma_{x'} A l'_1 + \rho_{x'} \sigma_{sx'} A l'_1 + \tau_{x'y'} A m'_1 + \tau_{x'z'} A n'_1 = \sigma_{nn} A l'_1 \tag{4.19}$$

$$\sigma_{y'} A m'_1 + \rho_{y'} \sigma_{sy'} A m'_1 + \tau_{x'y'} A l'_1 + \tau_{y'z'} A n'_1 = \sigma_{nn} A m'_1 \tag{4.20}$$

$$\sigma_{z'} A n'_1 + \tau_{y'z'} A m'_1 + \tau_{x'z'} A l'_1 = \sigma_{nn} A n'_1 \tag{4.21}$$

where (l'_1, m'_1, n'_1) , (l'_2, m'_2, n'_2) and (l'_3, m'_3, n'_3) are the direction cosines of the three principal stress directions with respect to the three reinforcement directions. The z' direction has been assumed to be unreinforced for the purpose of this discussion. When a small increment of load is applied on this element cracks are formed.

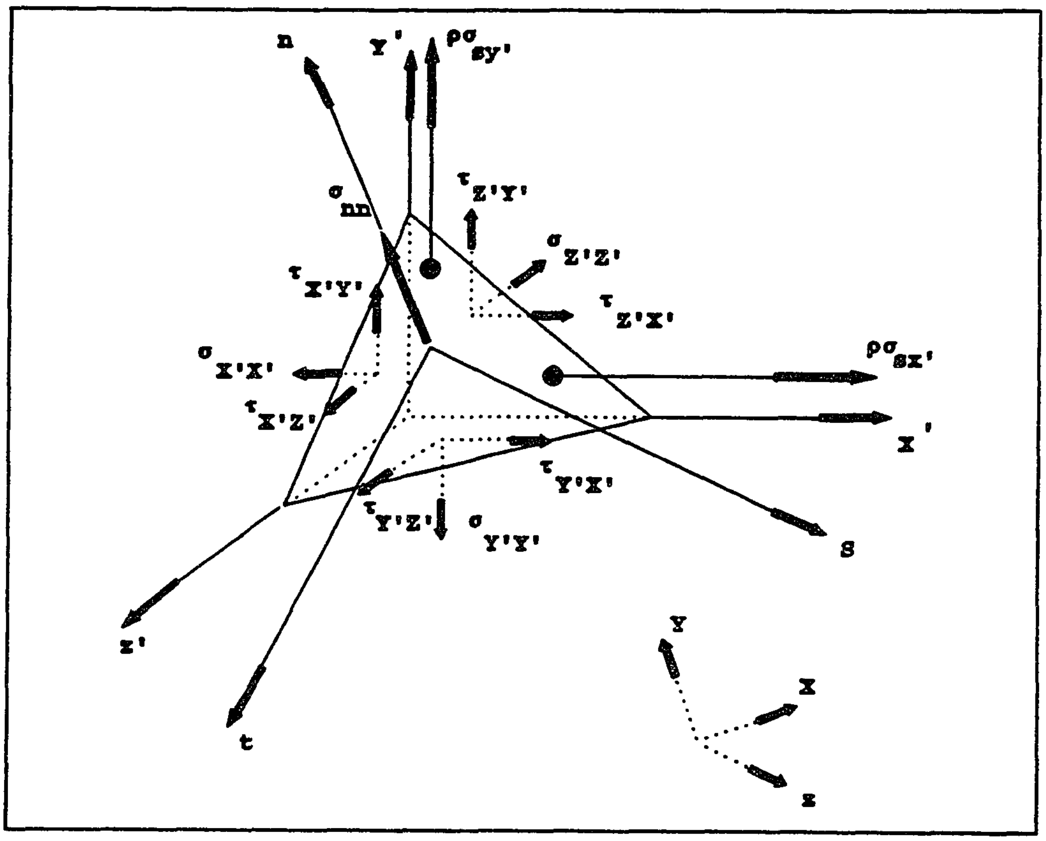


Figure 4.11: Bi-Directionally Reinforced Concrete Element Subjected to General Loads Prior to Cracking.

The shear stresses on the crack plane are no longer negligible if there are unbalanced loads in the unreinforced direction. In the crack normal direction the strain softening spring is initiated.

As before the strains in each steel direction is monitored(figure 4.12). When the strains in any reinforced direction reach the uniaxial cracking strain level the strain-softening in the crack normal direction is released. The tension-stiffening in each reinforcing direction sustains the forces released from the crack normal direction (figure 4.13).

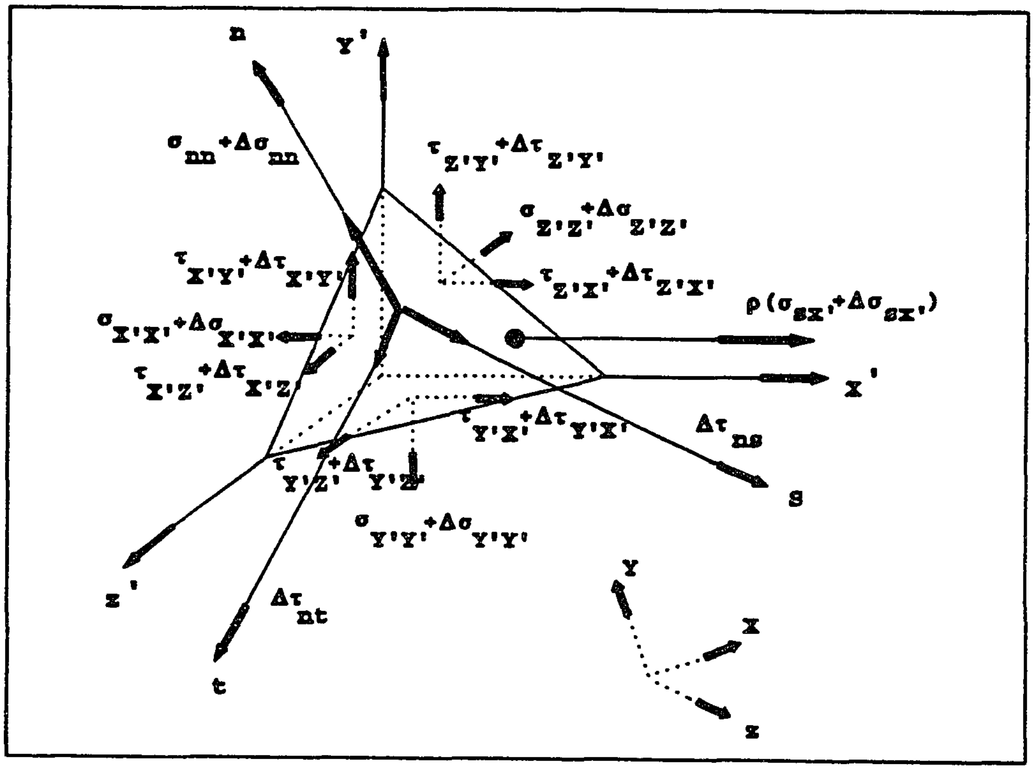


Figure 4.12: Bi-Directionally Reinforced Concrete Element Subjected to General Loads After Cracking, With Strain $\epsilon < \epsilon_{cr}$ in the Steel Direction.

An inherent assumption made here to compute the force release in each steel direction is that the contribution of the bond in each direction is the same ($\Delta\sigma'_x =$

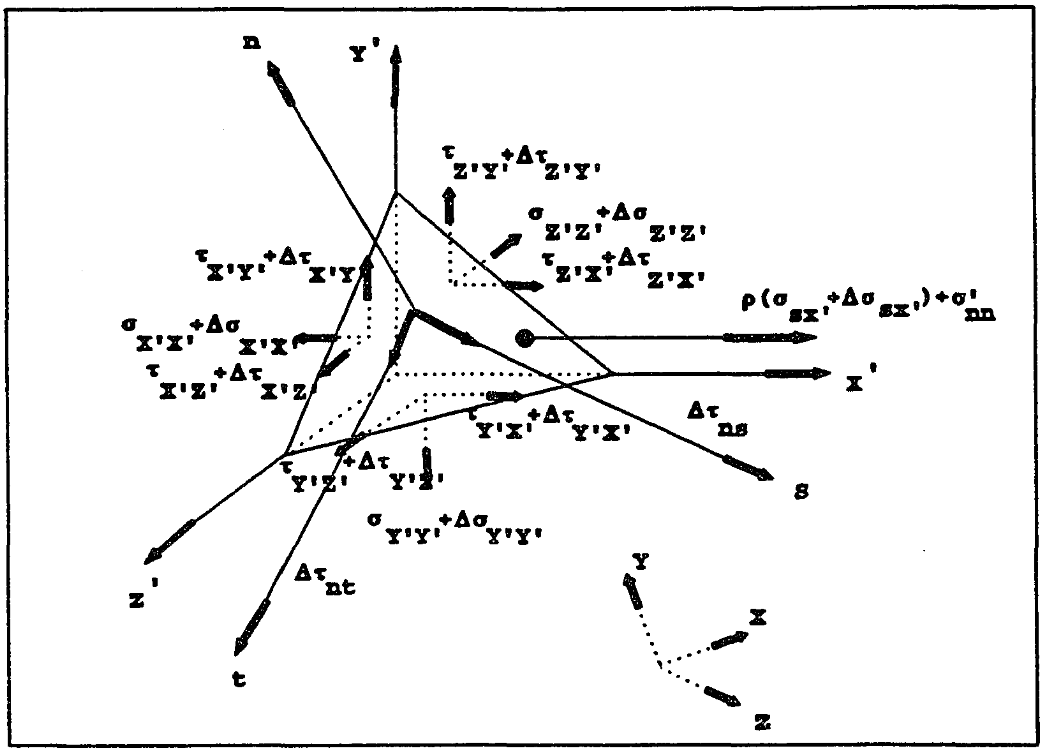


Figure 4.13: Bi-Directionally Reinforced Concrete Element Subjected to General Loads After Cracking, With Atleast One Strain $\epsilon \geq \epsilon_{cr}$ in the Steel Direction.

$\Delta\sigma'_y$). The increase in the stresses in each reinforced direction is:

$$\Delta\sigma'_x = \Delta\sigma'_y = \Delta\sigma'_n = \\ \Delta\sigma_n\{\frac{(l'_3n'_1 - l'_1n'_3)(m'_2n'_3 - m'_3n'_2) - (n'_1m'_3 - m'_1n'_3)(l'_2n'_3 - n'_2l'_3)}{n'_3[m'_1(l'_2n'_3 - n'_2l'_3) - l'_1(m'_2n'_3 - m'_3n'_2)]}\} \tag{4.22}$$

4.3.2.3 Uni-Directionally Reinforced Element

Consider next an element that has reinforcement only in one direction. When such an element is subjected to a general state of loads and it cracks, shear stresses develop very rapidly on the crack surface as this is a case of extreme anisotropy. Considering the element to be reinforced only in the x' direction the equilibrium equations at incipient cracking in local steel direction (x', y', z') are (figure 4.14):

$$\sigma_{x'}Al'_1 + \rho_{x'}\sigma_{sx'}Al'_1 + \tau_{x'y'}Am'_1 + \tau_{x'z'}An'_1 = \sigma_{nn}Al'_1 \tag{4.23}$$

$$\sigma_{y'}Am'_1 + \tau_{x'y'}Al'_1 + \tau_{y'z'}An'_1 = \sigma_{nn}Am'_1 \tag{4.24}$$

$$\sigma_{z'}An'_1 + \tau_{yx'z'}Am'_1 + \tau_{x'z'}Al'_1 = \sigma_{nn}An'_1 \tag{4.25}$$

With a small increment of load, cracks are formed and shear stresses begin to develop on the crack surface. Figure 4.15 shows the state of stress at this stage.

The strain in the steel direction are monitored till it reaches the uniaxial cracking strain, before releasing the strain-softening response in the crack normal direction. Immediately after the strain softening is released, the unreinforced direction becomes weak and shear stresses grow very rapidly on the crack plane to equilibrate the applied forces in the unreinforced direction (figure 4.16). The increase in stress along the steel direction obtained from equilibrium of forces on the crack plane is expressed as:

$$\Delta\sigma'_x = \Delta\sigma'_n = \\ \Delta\sigma_n\{\frac{(m'_2n'_1 - m'_1n_2)(m'_2l'_3 - m'_3l'_2) - (l'_2m'_1 - m'_2l'_1)(n'_2m'_3 - n'_3m'_2)}{l'_1m'_2(n'_2m'_3 - m'_2n'_3)}\} \tag{4.26}$$

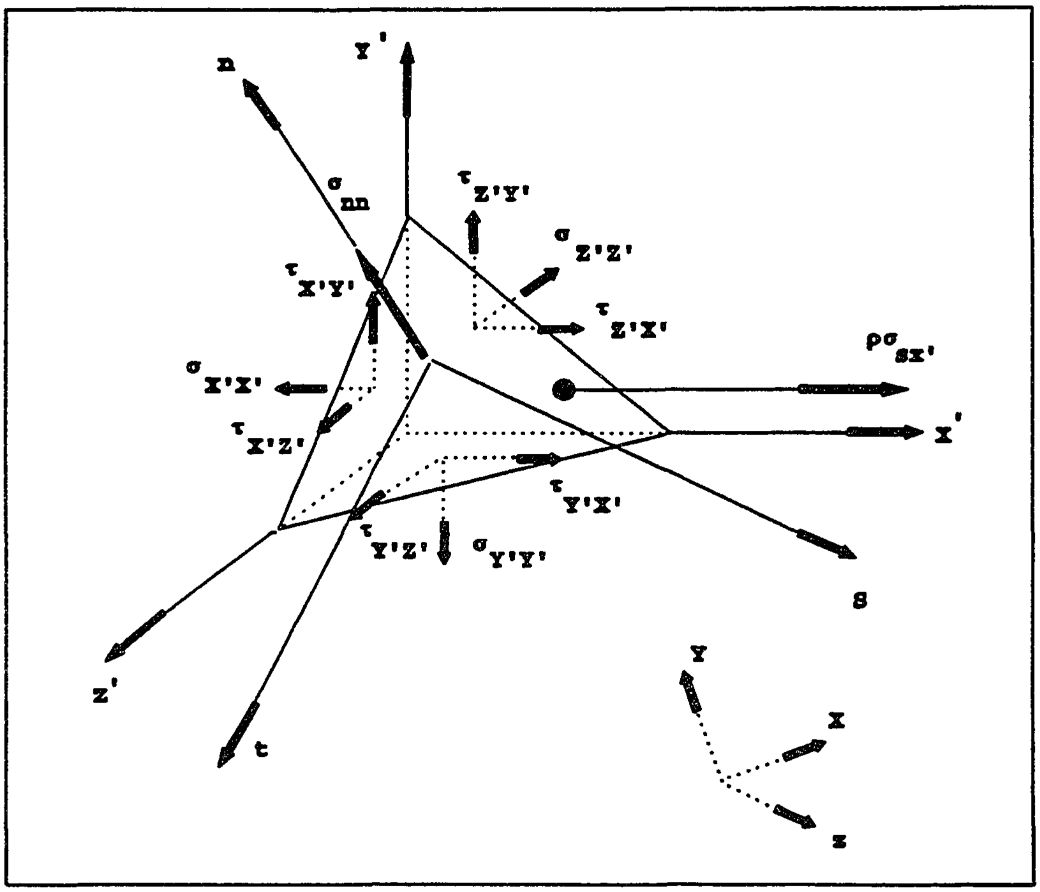


Figure 4.14: Uni-Directionally Reinforced Concrete Element Subjected to General Loads Prior to Cracking.

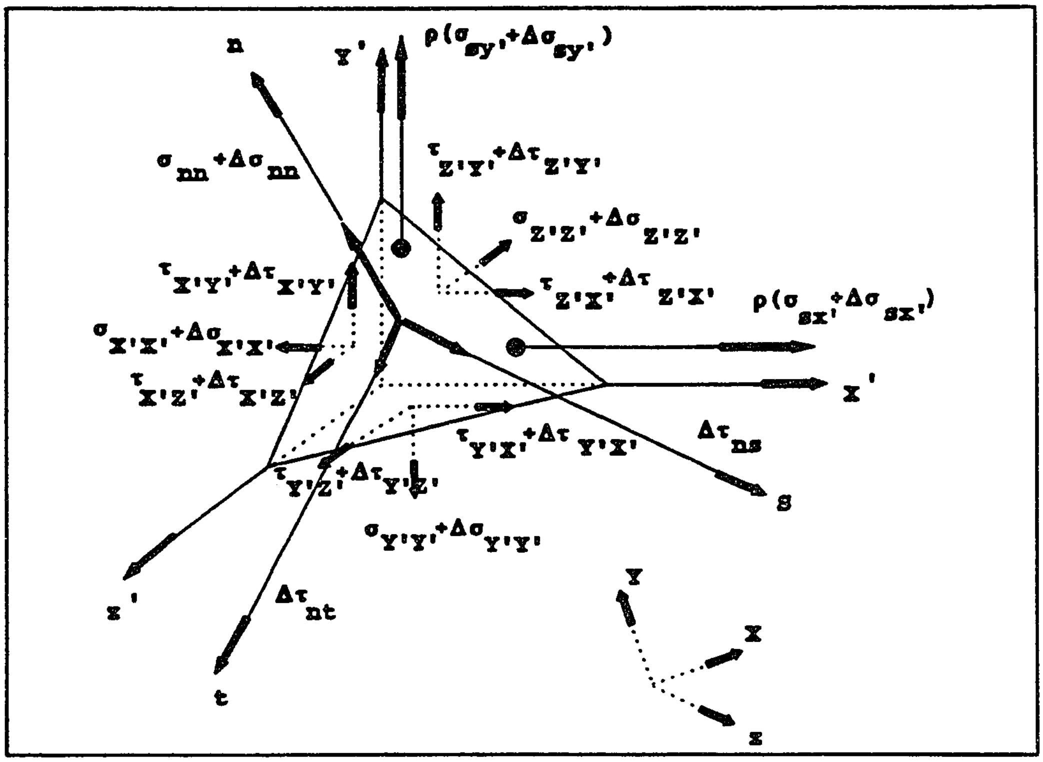


Figure 4.15: Uni-Directionally Reinforced Concrete Element Subjected to General Loads After Cracking, With Strain $\epsilon < \epsilon_{cr}$ in the Steel Direction.

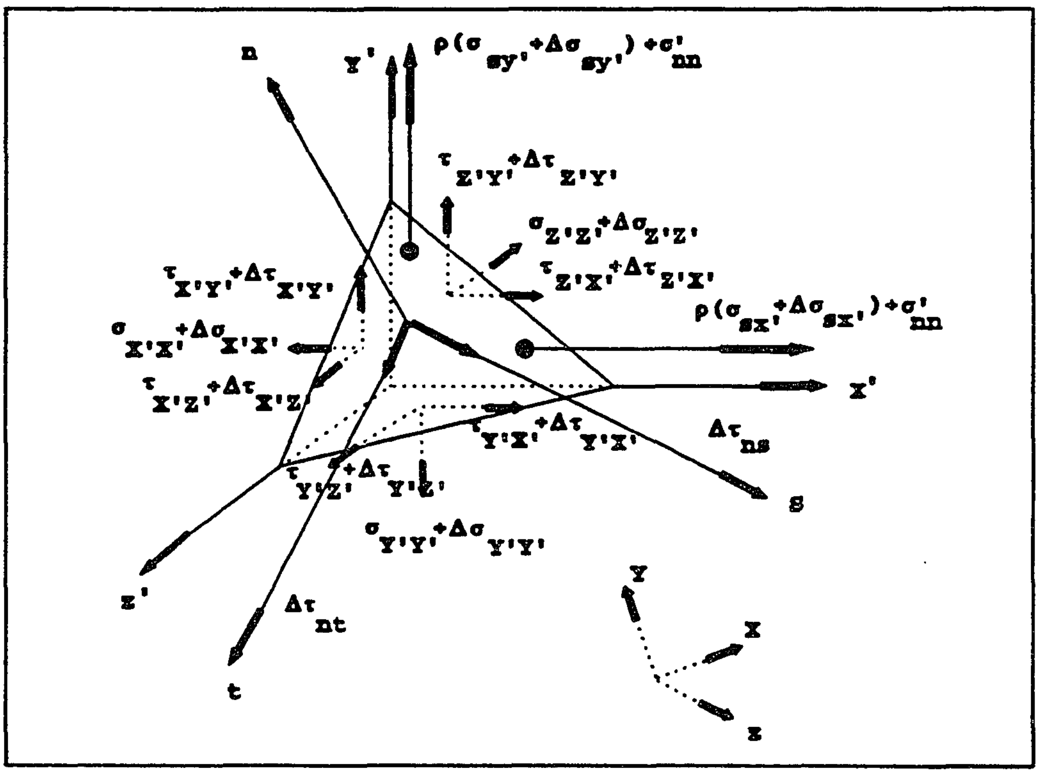


Figure 4.16: Uni-Directionally Reinforced Concrete Element Subjected to General Loads, With Strain $\epsilon \geq \epsilon_{cr}$ in the Steel Direction.

Thus the bond released to the reinforcement in such specimen is a function of the orientation of the crack with respect to the reinforcement. It should be pointed out that if the crack is inclined to the reinforcement beyond a certain threshold angle there will be some unbalanced shear forces left on the crack plane and equilibrium is unattainable through straining of the reinforcement. Amongst the panels tested by Bhide and Collins (1986), panel PB18 (results of which are presented in chapter 6) shows rapid failure immediately after the initial cracks form.

The response of such specimen is dependent on not only the bond but also the shear stiffness mobilized on the crack plane. As the loading increases the shear stresses on the crack surface grow rapidly. Additional cracking is possible if the stresses build up to the cracking state again. These subsequent cracks are treated as brittle and the previously established tension stiffening spring are not altered. If such cracks form oriented beyond an angle of 45° with all the reinforcing direction(s) these cracks are treated as strain-softening cracks in plain concrete.

4.3.3 Compression Softening of Cracked Reinforced Concrete

Experimental investigations (Vecchio and Collins 1982, Kolleger 1988, and Dyngland 1989) have indicated that after cracking there is a significant reduction in the concrete compressive strength and stiffness parallel to the cracks. This is especially true when the amount of reinforcement is high and consequently crushing of concrete precedes yielding of reinforcement. Some of the factors affecting the behavior of cracked compressive struts are (1) cracking pattern (2) crack spacing (3) crack width (4) reinforcement direction with respect to the crack plane and (5) the amount of reinforcement. Spacing of cracks and the irregularity in their shapes may lead to eccentric loads and stress concentrations in the compressive strut, thus reducing its strength. Additionally, a skewed orientation of the reinforcement with respect to the crack direction may result in 'tearing' out of the bar at the crack surface and formation of secondary cracking. This causes severe local deterioration of the concrete.

Based on the observations in their tests, Vecchio and Collins (1982, 1986) have proposed an expression relating compressive strength of cracked concrete σ^{co}_{max} to average tensile strains ϵ''_x normal to the compressive strut as (figure 4.17):

$$\sigma^{co}_{max} = \frac{f'_c}{0.80 - 0.34\frac{\epsilon''_x}{\epsilon'_c}} \leq f'_c \tag{4.27}$$

The uniaxial compressive strength of concrete is f'_c and the corresponding com-

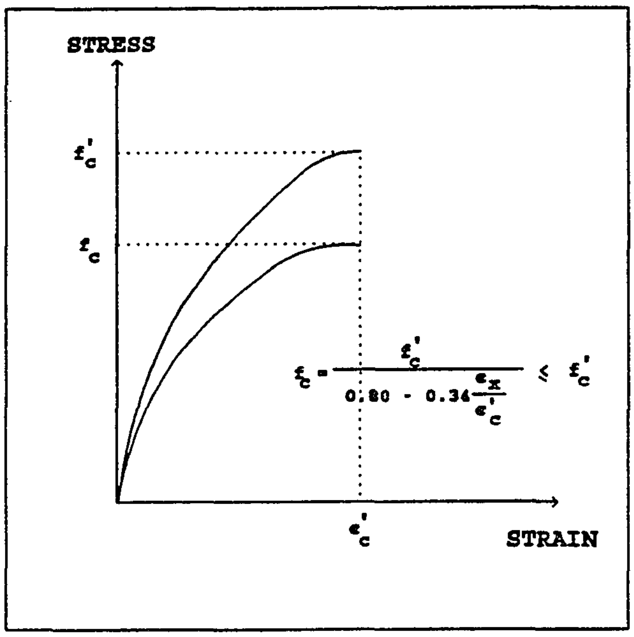


Figure 4.17: Compression-Softening in Cracked Reinforced Concrete, Vecchio and Collins (1982)

pressive strain is ϵ'_c. The stress-strain relation in the strut is given by:

$$\sigma^{co}_y = \sigma^{co}_{max}[2\{\frac{\epsilon''_y}{\epsilon'_c}\} - \{\frac{\epsilon''_y}{\epsilon'_c}\}^2] \tag{4.28}$$

ϵ''_x and ϵ''_y are the cracked concrete tensile and compressive strains, respectively. In this study equation 4.28 has been adopted to compute the stiffness of 'intact' concrete E_{co} after cracking. The modulus of elasticity is continuously updated as:

$$E_{co} = \frac{\sigma^{co}_y}{\epsilon''_y} \tag{4.29}$$

During analysis, the cracked concrete tensile strain (ϵ_x'') and compressive strain (ϵ_y'') are replaced by the maximum and minimum total principal strains, respectively. It should be pointed out that equation 4.28 was developed by (Vecchio and Collins 1982, 1986) for a particular crack orientation, while in this study it has been employed to compute the stiffness of the intact concrete (E_{co}) in any direction. But, as discussed in section 4.3.1, there is a rapid loss of stiffness in the crack normal direction due to the strain softening effect. Thus, the overall effect of using equation 4.28 is one of softening only the cracked concrete compressive strut, as proposed by Vecchio and Collins (1982,1986).

While most investigators accept the reduction in compressive strength for fact, there have been questions as to the level of this reduction. Both Kolleger (1988) and Dyngland (1989) have conducted tests on RC panels and concluded that equation 4.28 overestimates the strength reduction. In fact, they propose an upper limit of 20% to this reduction.

In this study another method to account for reduction in compressive strength of cracked concrete has been proposed. Upon cracking, the uniaxial compressive strength of concrete f_c' has been reduced linearly as:

$$f_c = f_c'(1.0 - 0.2\frac{\epsilon_{max}}{\epsilon_c'}) \;\geq\; 0.8f_c' \tag{4.30}$$

where ϵ_{max} is the current maximum principal tensile strain in concrete. f_c is the modified uniaxial compressive stress in concrete and ϵ_c' is the uniaxial compressive strain in concrete at peak compressive stress. Ottosen's triaxial strength envelope is modified to account for the change in the concrete compressive strength (figure 4.18). The corresponding strain at uniaxial compressive strength is also modified to account for the damage, due to cracking, and is expressed as:

$$\epsilon_c = \epsilon_c'(1.0 + 0.1\frac{\epsilon_{max}}{\epsilon_c'}) \tag{4.31}$$

Numerical studies of test specimens using each of the above formulations for compressive strength of cracked concrete are presented in Chapter 6.

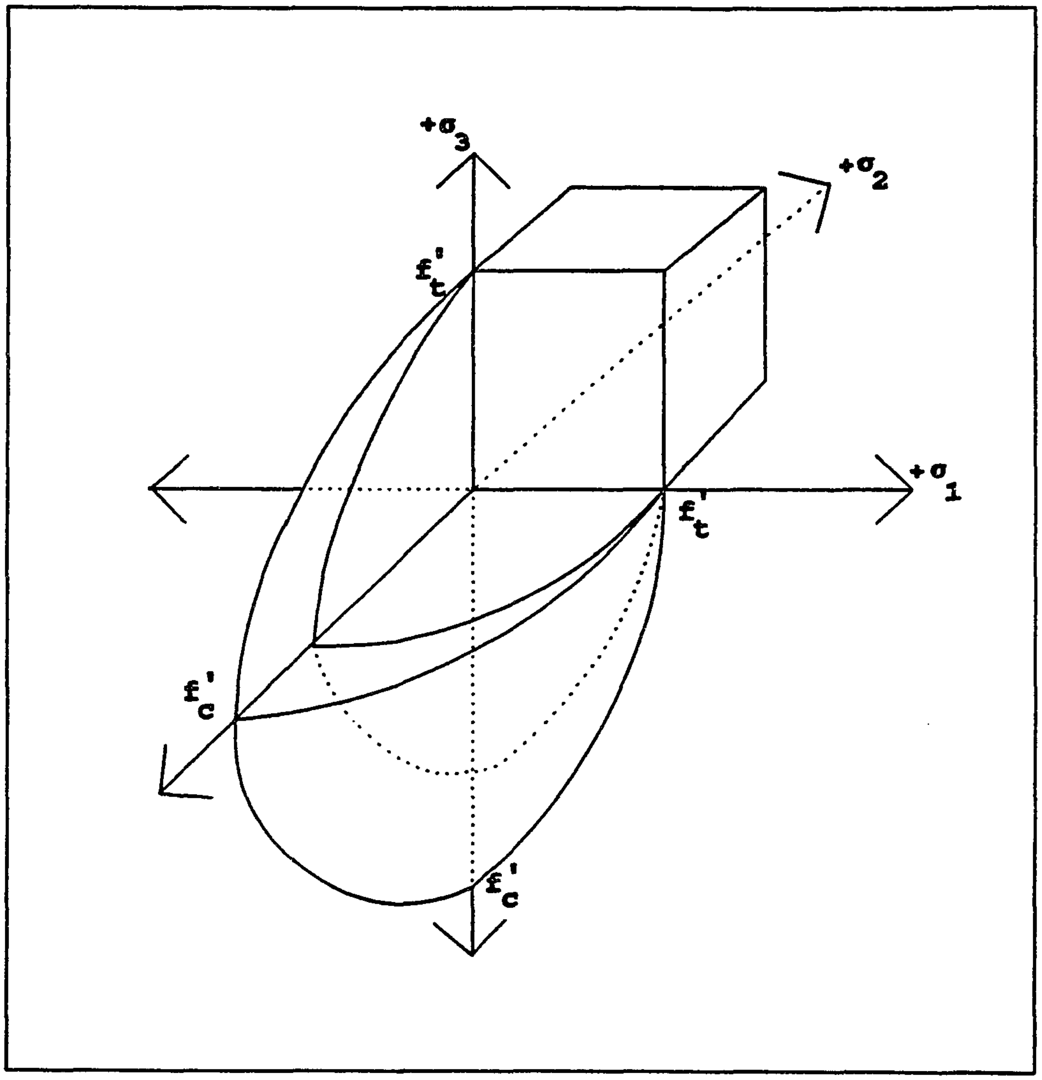


Figure 4.18: Compression-Softening in Cracked Reinforced Concrete, Present Study

4.3.4 Crack Interface Shear Transfer

Crack interface shear stiffness has been modeled in many studies by retaining a fixed fraction of the uncracked concrete shear stiffness as the crack shear stiffness in the crack stiffness matrix (equation 4.1).The shear stiffness of plain concrete elements is reduced to 1% of its value prior to cracking in this study. In RC elements, a small fraction of the uncracked concrete shear stiffness (20 to 50 %) has been found to be sufficient, when these elements are reinforced in at least two orthogonal directions. In uni-directionally reinforced elements, (e.g. Bhide and Collins 1987) such an approximation of the shear stiffness leads to a very stiff response and a more sophisticated formulation to describe this phenomenon is necessary.

Experimental evidence (Millard and Johnson 1984, 1985; Vintzeleou and Tassios 1987) indicates that shear slip is affected not only by the corresponding shear stress but also by the associated crack opening. Likewise, the crack normal stress is affected by the crack opening and the shear stress, especially in reinforced concrete where the reinforcement constrains the crack opening. Attempts to include this coupling effect in analyses have met with difficulty, as these coupling components have been found to be unequal rendering the matrix indefinite and even ill-conditioned (Yoshikawa 1989). Therefore, in the present study an attempt has been made to include the coupling effects implicitly in the interface shear stiffness, using experimental data.

Millard and Johnson (1984, 1985) have conducted tests on aggregate interlock specimen while considering both reinforced concrete and externally held cracked concrete specimen. They have concluded that the initial axial stiffness normal to the plane of the crack which restrains crack widening is up to five times higher when specimen are reinforced in these tests because of local bond. However the stiffness diminishes towards that of an unbonded bar as the tension stiffening deteriorates at higher loads.

Similar studies have been conducted by Tassios and Vintzeleou (1987). In their tests on aggregate interlock behavior, pre-cracked concrete specimen were held against each other at constant stress normal to the crack plane and then sheared in a direction parallel to the crack plane (figure 4.19). Vintzeleou and Tassios (1987) have also conducted dowel tests of reinforcement bars across frictionless concrete interfaces without any lateral confinement. These tests correspond to the externally held specimen tested by Millard and Johnson.

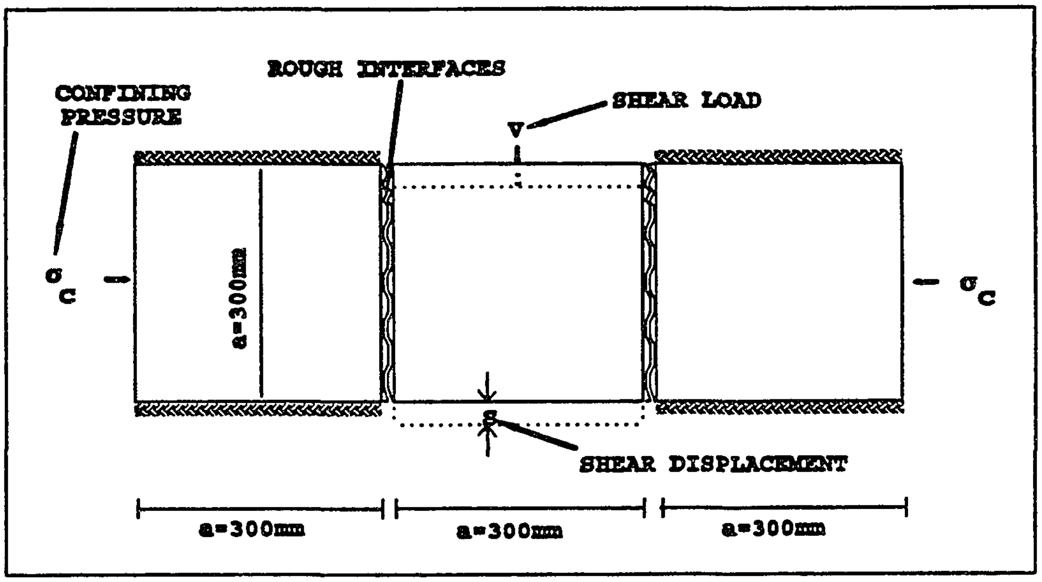


Figure 4.19: Experimental Setup Used to Study Crack Interface Shear Load vs. Displacement Behavior Under Lateral Confinement (σ_c), Tassios and Vintzeleou (1987).

In this study the results of the Tassios-Vintzeleou tests, (figure 4.20) representing crack interface shear stress-shear strain under different confining stresses, have been used to obtain the crack secant shear stiffness of cracked reinforced concrete elements. The bond stress is taken as a measure of the confinement imposed by the reinforcement across cracks. For any given crack, the crack shear strain and the crack normal bond stress are used to obtain the crack shear stiffness G_{TV}. As the bond between steel and concrete (tension-stiffening) is lost due to straining the crack shear stiffness decreases (aggregate interlock decreases) to that obtained from the dowel tests, i.e., no confinement ($\sigma_c = 0.0$ MPa). It is expected that after yielding the confinement at the crack surface will be significantly reduced. But, the experimental data employed in this study is only for unyielded dowels. Therefore, the shear stiffness obtained from the experimental data is reduced using a relative stiffness factor δ which represents the ratio of the current cracked reinforced concrete stiffness normal to the crack to its value at cracking. The interface shear stiffness used in equation 4.1 is then given by:

$$G_{cr} = \delta \ G_{TV} \tag{4.32}$$

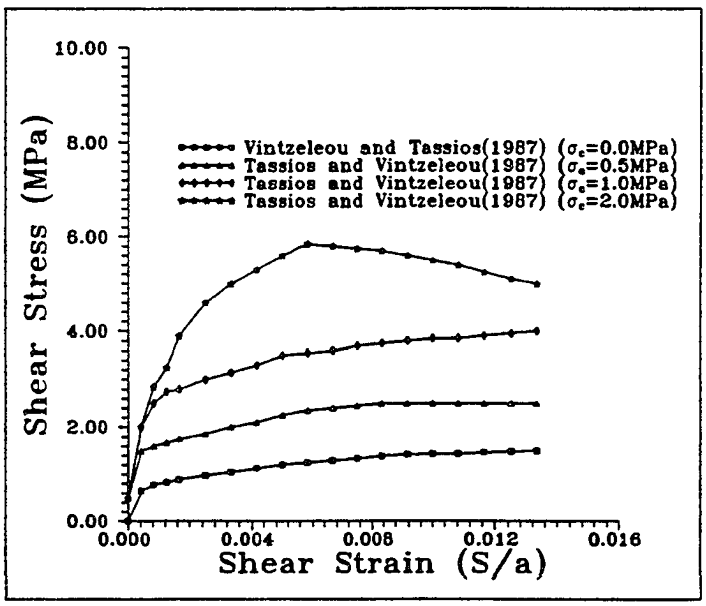


Figure 4.20: Crack Interface Shear Stress-Strain Curves at Various Levels of Confining Stresses Due to Aggregate Interlock and Dowel Action, Tassios and Vintzeleou (1987).

For numerical reasons, the minimum crack shear stiffness retained is 1% of the shear stiffness calculated for intact concrete at cracking.

Chapter 5

Numerical Implementation Aspects

5.1 Introduction

The three dimensional constitutive model for Reinforced concrete discussed in Chapter 3 and 4 has been implemented into a special purpose finite element program. For this purpose a finite element program used for studying metal forming processes, Foroozesh (1989), has been used as a base program for implementing the constitutive model for RC, layering procedures, numerical (secant) algorithm and the material point convergence procedures employed. For pre- and post-processing of the results the program is interfaced with the 'CAEDS', finite element graphics package available on the IBM 3090 system. The details of these procedures are described in this chapter.

5.2 Simulation of Concrete

In this study, for the simulation of concrete, a twenty noded isoparametric solid element and a eight noded degenerate shell element (Ahmad et al. 1970) have been employed. The solid element has been used primarily for ratifying the implementation of the constitutive model for concrete (Details of the material model ratification have been presented in section 3.2.3). The eight noded degenerate shell element has six degrees of freedom at each node: three translational and three rotational (figure 5.1). The 3-D stress-strain relationships for concrete given by equation 3.6 is condensed by enforcing the shell constraint of no (shell) normal stresses ($\sigma_{z'} = 0.0$). Thus the stress-strain relations for concrete in the shell local coordinates (x', y', z') is given by:

$$\begin{pmatrix} \sigma_{x'} \\ \sigma_{y'} \\ \sigma_{z'} \\ \tau_{x'y'} \\ \tau_{y'z'} \\ \tau_{x'z'} \end{pmatrix} = \frac{E_{co}}{1-\nu_{co}^2} \begin{pmatrix} 1 & \nu_{co} & 0 & 0 & 0 & 0 \\ \nu_{co} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-\nu_{co}}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & K_1\frac{1-\nu_{co}}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & K_2\frac{1-\nu_{co}}{2} \end{pmatrix} \begin{pmatrix} \epsilon_{x'} \\ \epsilon_{y'} \\ \epsilon_{z'} \\ \gamma_{x'y'} \\ \gamma_{y'z'} \\ \gamma_{x'z'} \end{pmatrix} \qquad (5.1)$$

where E_{co} and ν_{co} are the secant stiffness and Poisson's ratio of the material respectively. K_1 and K_2 are the transverse shear energy correction factors in the $y'z'$ and the $x'z'$ planes respectively.

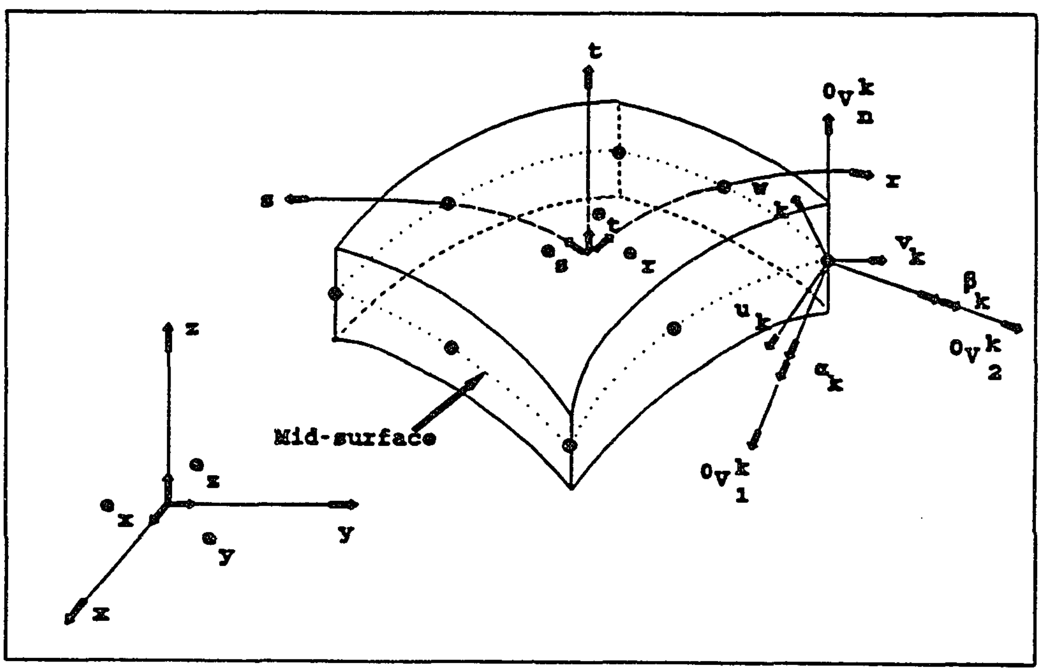


Figure 5.1: Eight-Noded Degenerate Shell Element.

In order to calculate the stiffness of the material accurately, a 3x3 full Gaussian integration is required in the plane of the element. However, the use of full integration has been shown to yield stiff responses (Zienkiewicz et al 1971) . This phenomenon occurs due to the extreme ratios of the bending stiffness to the shear and membrane stiffness. Shear locking has been shown to be suppressed with the use of reduced integration (2x2) of the shear terms (Zienkiewicz et al. 1971). In curved shell applications however it has been shown that the locking is due to the high ratios of the membrane stiffness to bending stiffness (Parisch 1979). The use of selective or uniformly reduced integration eliminates this problem. In material nonlinear problems however, the use of selectively reduced integration introduces difficulties in choosing the most optimal points for computing the stresses and strains for the different (bending, shear, membrane) stiffness components.

The use of uniformly reduced integration on the other hand, results in rank deficiencies in the stiffness matrix and the possibility of activating zero energy modes. A number of stabilization procedures have been proposed (eg. Belytchko and Tsay 1983, Milford and Schnobrich 1984) to suppress this rank deficiency. In the present study the full 3x3 and fully reduced 2x2 Gaussian integration procedures in the plane of the element have been implemented. In the shell thickness direction numerical integration is carried out using one, two or three sample points, located at 'Gauss points' in the thickness direction.

5.3 Simulation of Reinforcement

The reinforcement is treated as a smeared membrane layer with uniaxial stiffness properties oriented along the direction of the steel bars. The thickness of the reinforcement layer within an element is determined such that the cross-section of the reinforcement in the given element volume corresponds to the actual steel cross-section within the same volume. In this study a three dimensional degenerate shell element is employed for simulating the steel. A 3x3x1 Gaussian numerical integration procedure is employed in computing the stiffness of the steel.

5.4 Simulation of Cracks

A smeared cracking procedure has been employed to simulate a crack in this work. Here, the cracked solid is considered as a continuous medium and the cracks are modeled by means of inelastic strains (the rheological representation of cracks has been discussed in chapter 4). With such a treatment the effect of each crack is spread over a portion of the element.

A gradual reduction of the tensile stresses transferred at a cracked point causes a redistribution of stresses in the region. As discussed in section 2.4.2.1, a common feature of models employing the smeared crack procedure is their lack of objectivity in simulating crack propogation with respect to refinement of the finite element mesh (Bazant and Cedolin 1979, 1983). This objectivity with respect to mesh refinement can be achieved by modifying the constitutive law and making the fracture energy of concrete depend on the mesh size by introducing a parameter called 'crack band width' (Bazant and Oh, 1983). In regular meshes this parameter can be determined intuitively, but for irregular meshes and cracks skewed to the element sides this is no longer possible. However, there are some limitations. First, there is an upper limit for the crack band width (Bazant and Oh, 1983). In the analysis of large structures this limit may be exceeded and the introduction of a mesh dependent strength limit becomes attractive (Bazant and Cedolin 1979, 1983). Secondly, Bazant (1984) and Bazant, Belytschko and Chang (1984) have stated there is a lower limit for the crack band width, which roughly equals about three times the size of the aggregate. They pointed out that this lower limit has to do with the fact that the present formulation of the crack model is based on the local continuum theory. This theory would be unable to produce a detailed resolution of the stress and strain fields within and near the strain-softening region. Therefore, the size of an element within a finite element mesh should be greater than three times the size of the aggregate.

Oliver(1989) has proposed a method of evaluating the band width at cracked integration point which has been shown to be objective with mesh refinement. In this study this procedure has been employed.

For the simulation of more than one crack at a given sample point (non-orthogonal cracking) it has been found the strength criteria for crack initiation by itself is insufficient. It is quite possible that a series of cracks can be initiated resulting in 'numerical cracking' at that point. Thus, in this study the strength criteria is augmented by a minimum threshold angle criterion, whereby a new crack is initiated only if the strength criterion is violated and the crack normal of the new crack is directed at a skew larger than the threshold angle with respect to the normals of any previously existing cracks. Because such a criterion may cause the tensile stresses to build up to levels greater than the cracking strength of concrete under uniaxial tension, a tension-cutoff procedure is employed limiting the tensile stresses at a sample point to the uniaxial cracking strength of concrete. Typically the threshold angle used ranges from 15 to 30 degrees. However, it has been observed (e.g. de Borst and Nauta 1985) that for some problems this range is not sufficient and angles as large as 60 degree have been employed.

5.5 Layering Procedures

In this study layering procedures have been employed to capture the variation of material properties and the propagation of cracks through the thickness of the structure. In this study, two formulations have been used: an implicit layering procedure (Figueras and Owen 1983) and an explicit layering procedure (Barzegar 1989).

5.5.1 Implicit Layering Procedure

In the implicit layering procedure each finite element of the structure is made up of layers (figure 5.2). The constitutive aspects of the material are considered only at the layer level and element response is obtained from equilibrium and compatability considerations. At layer interfaces, continuity of both the transverse shear stresses and displacements is required. Assuming that the 'normals' to the middle surface of the shell remain straight but do not necessarily remain normal to the middle surface of the shell, the latter continuity is satisfied but the former is not. This assumption makes the transverse shear strain constant through the element thickness, which is a coarse approximation to the actual variation, even for homogeneous cross-sections. For homogeneous cross-sections, the transverse shear stress distribution is commonly accepted to be a parabolic function of the thickness coordinate (z). Therefore, a correction factor K must be introduced inorder to approximate , on an average basis, the transverse shear strain energy. For homogeneous layers a correction factor $K_1 = K_2 = 5/6$ is used in computing the strain energy. In the case of heterogeneous layers this value is no longer accurate. This is a limitation since these shear stress components cannot be used in nonlinear material model parameter computations. Figueras and Owen (1983) have proposed a more accurate method for computing shear strain energy correction factors for heterogeneous cross sections and have also suggested a method of obtaining shear stress components that are more accurate. They have employed this procedure in studying the behavior of orthotropic composite plates and have employed a plasticity-based material model description for this purpose. A brief description of this method (Figueras and Owen, 1983) follows.

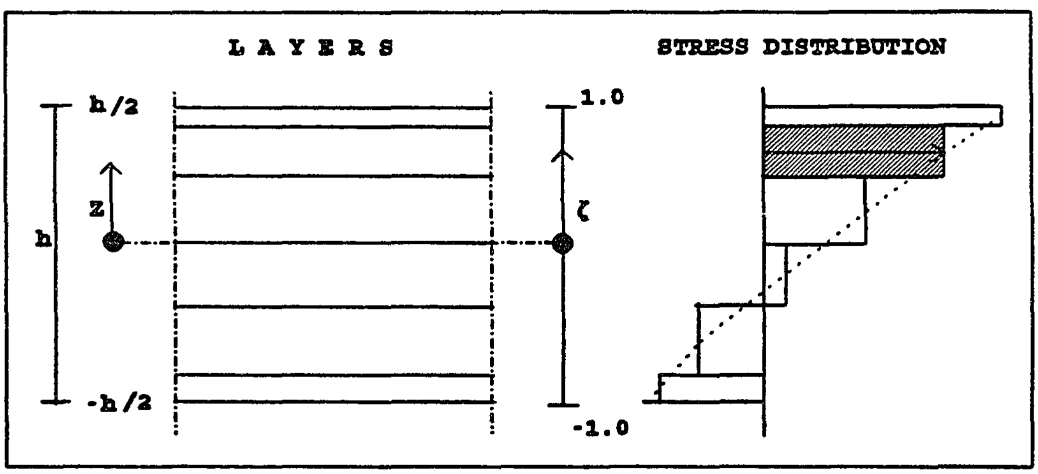


Figure 5.2: Implicit Layering Procedure

The basic assumptions used by them in the development of this formulation are those of cylindrical bending and traction free surfaces. For a plate of thickness 'h', setting z = h/2 at the top surface and z=-h/2 at the bottom surface the transverse shear stress in the xz plane (obtained from equilibrium considerations) is:

$$\tau_{xz} = -\int_{-h/2}^{z} \frac{\partial \sigma_x}{\partial x} dz = -\frac{Q_x}{R_1}\, g(z) \tag{5.2}$$

where:

Q_x is the shear force on the xz plane;

$R_1 = \int_{-h/2}^{h/2} D_1(\overline{z})\, \overline{z}^2\, d\overline{z}$ is the flexural plate stiffness in the x direction;

$\overline{z}$ is a variable coordinate through the thickness;

$g(z) = -\int_{-h/2}^{z} D_1(\overline{z})\, \overline{z}\, d\overline{z}$ is the shear stress distribution function.

Expressing the strain energy component(per unit mid-plane surface) in terms of R_1 and g(z) :

$$w_s = \frac{Q_x^2}{R_1^2} \int_{-h/2}^{h/2} \frac{g^2(z)}{G_{13}(\overline{z})}\, d\overline{z} \tag{5.3}$$

where $G_{13}(\overline{z})$ is the variable shear modulus in the xz plane.

Similarly the shear strain energy under the constant shear strain assumption is:

$$\overline{w}_s = \frac{Q_x^2}{h\overline{G}_1} \quad \text{where} \tag{5.4}$$

$$h\overline{G}_1 = \int_{-h/2}^{h/2} G_{13}(\overline{z})d\overline{z}$$

The strain energy correction factor in the xz plane K_2 is obtained by equating these energies and is expressed as:

$$K_2 = \frac{\overline{w}_s}{w_s} = R_1^2\ [h\overline{G}_1 \int_{-h/2}^{h/2} \frac{g^2(z)}{G_{13}(\overline{z})}\, d\overline{z}\,]^{-1} \tag{5.5}$$

The shear correction factor K_1 in the yz plane is obtained in the same way. In addition they have proposed expressions for computing the transverse shear stresses that account for material inhomoginity. In the xz plane τ_{xz} is obtained as:

$$\tau_{xz}^{COR.} = G_{13}(\overline{z})\, \frac{g(z)}{\overline{g}} \tag{5.6}$$

where:

$$\overline{g} = \frac{1}{h} \int_{-h/2}^{h/2} g(z)d\overline{z} \tag{5.7}$$

With this procedure an improved shear stress distribution through the thickness is obtained. The shear stress in the yz plane is computed in the same way. For numerical implementation all integrations are performed over the layer thickness and then summed up to account for material inhomginity.

The strain energy correction factor K_1 and K_2 represent the effect of the shear stress distribution, and also accounts for the material inhomoginity, in an average sense over the entire cross-section of the element. However, in the present study a layerwise numerical integration procedure is being employed. Thus, in the present study the shear stress distribution function given by equation 5.6 is used. To approximately account for the stress distribution within each layer, a factor of 5/6 for homogeneous layers, has been augmented to the expression given in equation 5.6.

5.5.2 Explicit Layering Procedure

The explicit layering scheme proposed by Barzegar (1989) employs stacked elements for considering variation of material properties through the thickness. Each element is associated with explicit degrees of freedom, some of which are constrained. In this procedure the number of active degrees in a node stack is large. The displacement degrees of freedom in the lower 'layers' (elements) are explicitly constrained in terms of the degrees of freedom of the layer above as:

$$U_2 = U_1 \mp \phi_{1y} \frac{h_1}{2} \mp \phi_{2y} \frac{h_2}{2} \tag{5.8}$$

$$V_2 = V_1 \mp \phi_{1x} \frac{h_1}{2} \mp \phi_{2x} \frac{h_2}{2} \tag{5.9}$$

$$W_1 = W_2 \tag{5.10}$$

where h_1 and h_2 are the thicknesses of adjacent 'layers' (elements) in a stack and U_1, V_1, W_1, ϕ_{1x}, ϕ_{1y}, U_2, V_2, W_2 and ϕ_{2x}, ϕ_{2y} are the translational and rotational degrees of freedom of the corresponding nodes of adjacent elements in the stack. Figure 5.3 shows the details of this procedure. The primary advantage of this layering procedure is in its ability to capture the variation of the transverse shear strain and stress through the thickness accurately. However, computationally this procedure is expensive because of the increase in nodal degrees of freedom. The constraint equations generated by this layering procedure are implemented via a lagrange multiplier technique described in Cook(1981).

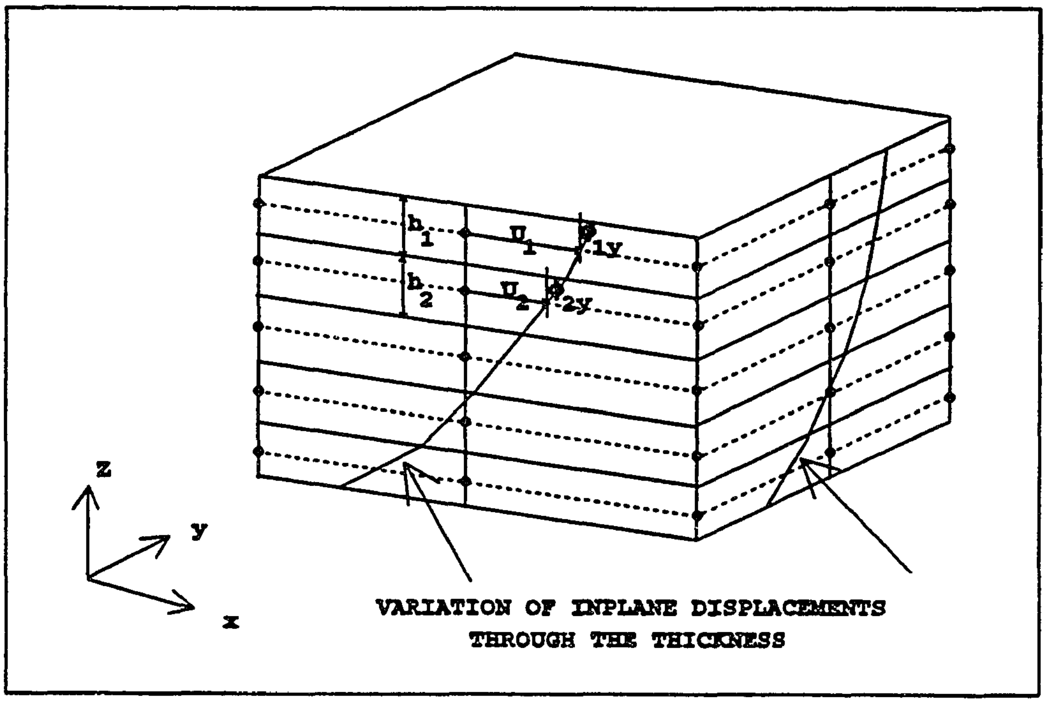


Figure 5.3: Explicit Layering Procedure

In treating reinforcement, some modification of the explicit layering procedure has been made in this study. The steel layer along with the concrete layer adjacent to it are grouped into a single element through the use of the implicit layering procedure.

5.6 Computational Scheme

The total secant finite element formulation used in this study requires the solution of the following equation for the displacements.

$$\{K\}\ [U] = [F] \tag{5.11}$$

Here $\{K\}$ is the global secant stiffness matrix, [F] is the total load vector and [U] is the total displacement vector. Details of formulating the stiffness matrix and load vector are presented in Bathe (1978). For nonlinear material problems the stiffness matrix $\{K\}$ depends on the stresses within the material and so an iterative solution procedure is necessary. Figure 5.4 illustrates the functioning of the secant algorithm and the path to convergence. The system of equations obtained in equation 5.11 are solved by the Gauss elimination method using the active column storage technique (Bathe 1982).

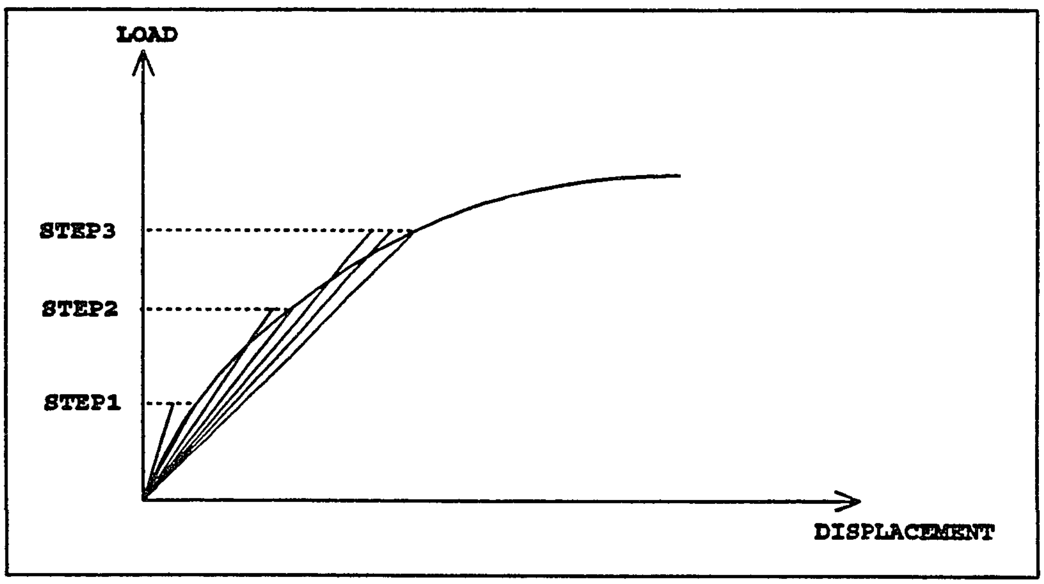


Figure 5.4: Secant Iterative Procedure.

5.6.1 Convergence Procedures

For a solution strategy based on iterative methods to be effective a realistic criteria should be employed for the termination of the iteration at a given load level. At the end of each iteration, the solution obtained should be checked to see if it has converged within the preset tolerances (3-5%) or if divergence was detected. If convergence tolerances are not strict, inaccurate results may be obtained, and if the tolerances are too strict, much computational effort is spent on needless accuracy.

The convergence criterion employed depends on the type of problem being examined. For example when the displaced configuration is being sought at a given load level, convergence is assumed if the displacement at the end of each iteration is within a specified tolerance of the displacement at the end of the previous iteration. However, in some problems the variation in the displacement may be very small within each iteration while the actual solution may be far from the one obtained using this procedure. In such cases a convergence criterion based on requiring the out of balance load vector at any iteration to be within a tolerance of the load increment may be more appropriate. In order to provide some indication of when both the displacement and the forces are near their equilibrium position a criterion based on energy is sometimes employed. Here convergence is assumed when the increment in internal energy (work done by the out of balance load vector in the previous iteration) is within a prescribed tolerance of the initial internal energy increment computed in the first iteration of the current load increment.

For the total secant stiffness formulation implemented in the present study an energy based criterion enforced at each integration point of each layer within each element is employed (Barzegar 1986). The details of its implementation in the present study is discussed in this section.

5.6.1.1 Intact concrete Response

Pre Peak Convergence

The displacements obtained from solution of equation 5.11 are processed at each sampling point to obtain the total strain components $\{\epsilon\}_{x,y,z}$ at these points. The total stresses $\{\sigma\}_{x,y,z}$ at these points are recovered using equation 4.9. The principal stresses are then computed. The principal strains in the uncracked concrete are computed using the principal stresses. If all these strain components are less than the corresponding largest strain components previously registered at this point the point is unloading and convergence is assumed. If any strain component is found to be greater than its previously stored value, loading is assumed and the convergence criterion described here is enforced. Figure 5.5 illustrates the convergence procedure employed in the pre-peak region at a Gauss point under a uniaxial state of stress. E_1 is the used secant stiffness for concrete in the current iteration resulting in a concrete strain ϵ_1. Using the calculated stress $\sigma_1 = E_1\epsilon_1$ a new stiffness E_1' is determined. If the difference between the used stiffness E_1 and the new stiffness E_1' is less than the tolerance (typically 3-5%), convergence of the stress-strain law is assumed. If this requirement is not satisfied then a new stiffness, E_2, is obtained as :

$$E_2 = \text{Minimum}(\frac{E_1 + E_1'}{2}, E_1(1 - \text{tolerance})) \tag{5.12}$$

As seen in figure 5.5 the convergence path is one of gradual softening at the Gauss point leading to release of stresses in the region.

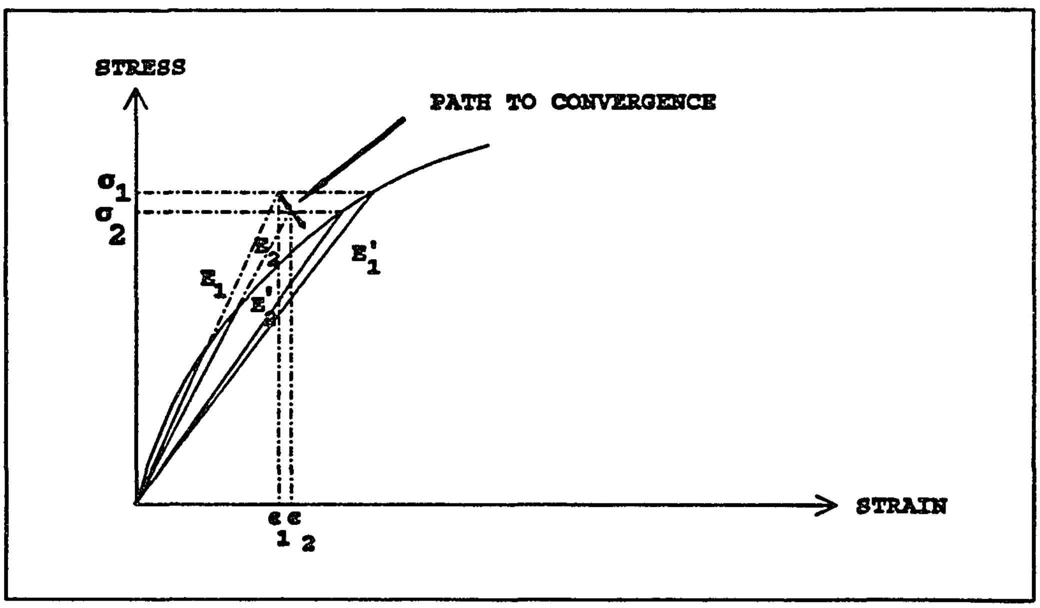


Figure 5.5: Convergence Path at a Point in Concrete Prior to Crushing.

The convergence procedure employed to modify the Poisson's ratio is similar to the one employed for concrete stiffness. When the used Poisson's ratio ν_1 is less than the updated one ν_2 by more than the specified tolerance, a new Poisson's ratio obtained as:

$$\nu_3 = \nu_1\,(1.0 + \text{tolerance}) \tag{5.13}$$

To detect limit points in the stress-strain curve the tolerances used are automatically reduced internally. Further, overshooting of the stresses at the Gauss point is also corrected by a gradual reduction of the stiffness.

Post-Peak Convergence

In the post-peak strain-softening zone similar modification of the secant modulus and Poisson's ratio are made (figure 5.6). If the used stiffness E_1 is less than its current value, E_1', by more than the specified tolerance then a new stiffness is required. The new secant stiffness employed is $E_2 = E_1(1 - \text{tolerance})$. As mentioned in section 3.3 the Poisson's ratio corresponding to E_2 is obtained as ν_2^{*} such that the bulk modulus at peak compressive stress remains unchanged. The new Poisson's ratio is then calculated as $\nu_2 = 1.005\nu_2^{*}$.

5.6.1.2 Crack Interface Response

At each sampling point the total global strain components $\{\epsilon\}_{xyz}$ are recovered from the nodal displacements. These strain components and the cracked concrete stiffness (equation 4.7) are used to obtain the total global stresses $\{\sigma\}_{xyz}$ at that sampling point. The crack stress components of each crack $\{\sigma\}_{nst}^{cr}$ are then obtained from the global stresses using the stress transformation matrix.

The current crack normal strain ϵ_1 is obtained from the crack normal stress σ_{nn} and the employed crack stiffness E_1 (figure 5.7). If this strain component is less than its largest value previously registered at this point the point is unloading and convergence is assumed. If the strain is found to be greater than the previously stored strain loading is assumed and a convergence criterion described here is enforced. The new stiffness E_2 is obtained from the crack strain ϵ_1 and the stress-strain relationship employed for the crack. The total area under the stress-strain

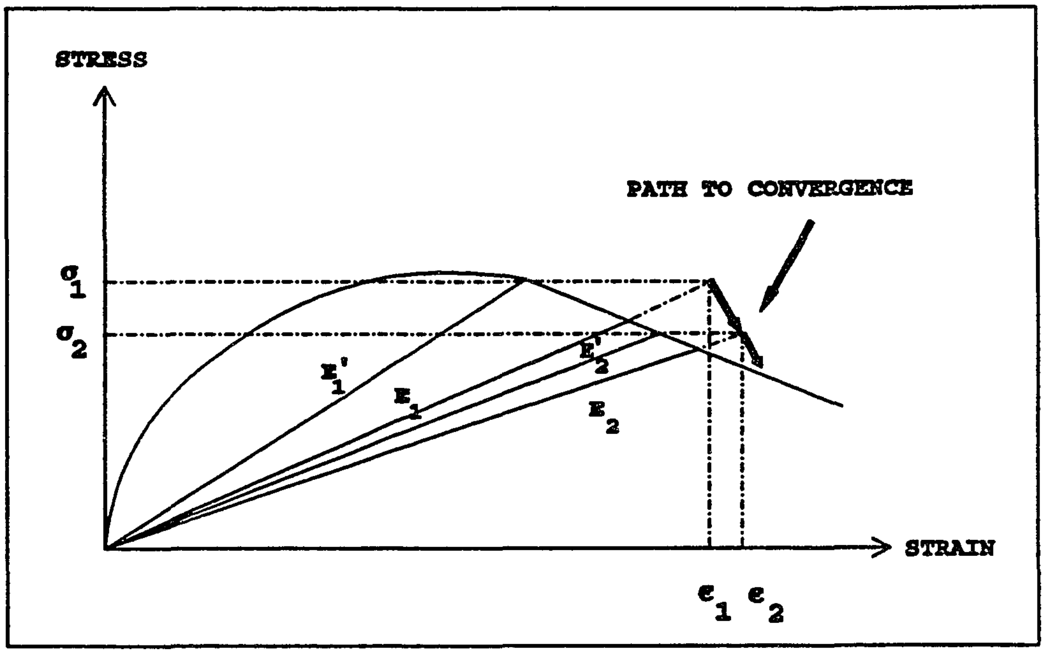


Figure 5.6: Post-Peak Convergence Path in Concrete.

diagram for the crack is A, while the area under the stress-strain curve between E_1 and E_2 is A_1. Convergence is assumed if the area A_1 in the current iteration satisfies the relationship:

$$A_1 \leq A * \text{ tolerance} \tag{5.14}$$

where the tolerance is typically 3-5%. If this not satisfied E_2 is assumed to be the new stiffness and further iterations are carried out.

Great care is needed in employing the above criterion in the initial stages of an opening crack (figure 5.8). The change in the stiffness over successive iterations is so small that equation 5.14 would indicate convergence. However, the stress point has violated the crack strength f_{ct} as seen in figure 5.8. In this situation the convergence is rejected and a new stiffness E_2' is employed. This stiffness E_2' is obtained such that the area A_2 under the stress strain curve between E_1 and E_2' is obtained as :

$$A_2 = \text{tolerance } * A \tag{5.15}$$

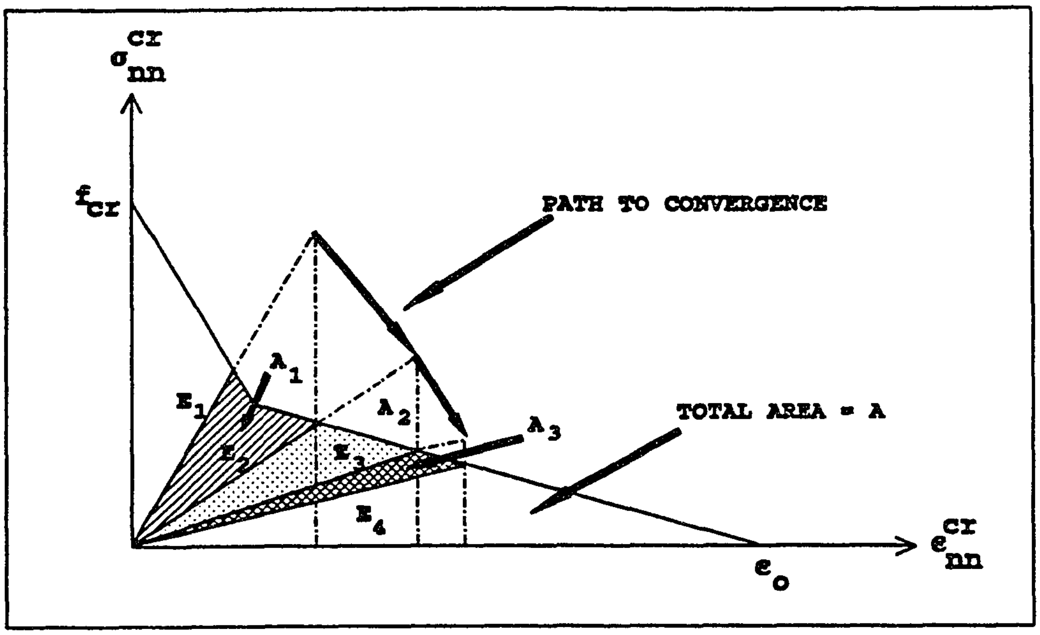


Figure 5.7: Convergence Path in a Strain Softening Cracked Point.

More iterations are carried out until the stress point satisfies the constitutive law at the point.

5.6.1.3 Tension-stiffening Response

The tension-stiffening strain ϵ_1 is obtained by transforming the global strains $\{\epsilon\}_{xyz}$ through the strain transformation matrix. If this strain component is less then its largest strain level previously registered at this point the point is unloading and convergence is assumed. If the strain is found to be greater then the previously stored strain, loading is assumed, and a convergence criterion described here is enforced. The tension-stiffening strain ϵ_1 is employed to compute the new stiffness E_2 using the stress-strain relationship (figure 5.9). The total area under the stress-strain diagram is A. The area under the stress-strain curve between E_1, the stiffness used in the current iteration, and E_2 is A_1. Convergence is assumed if the following

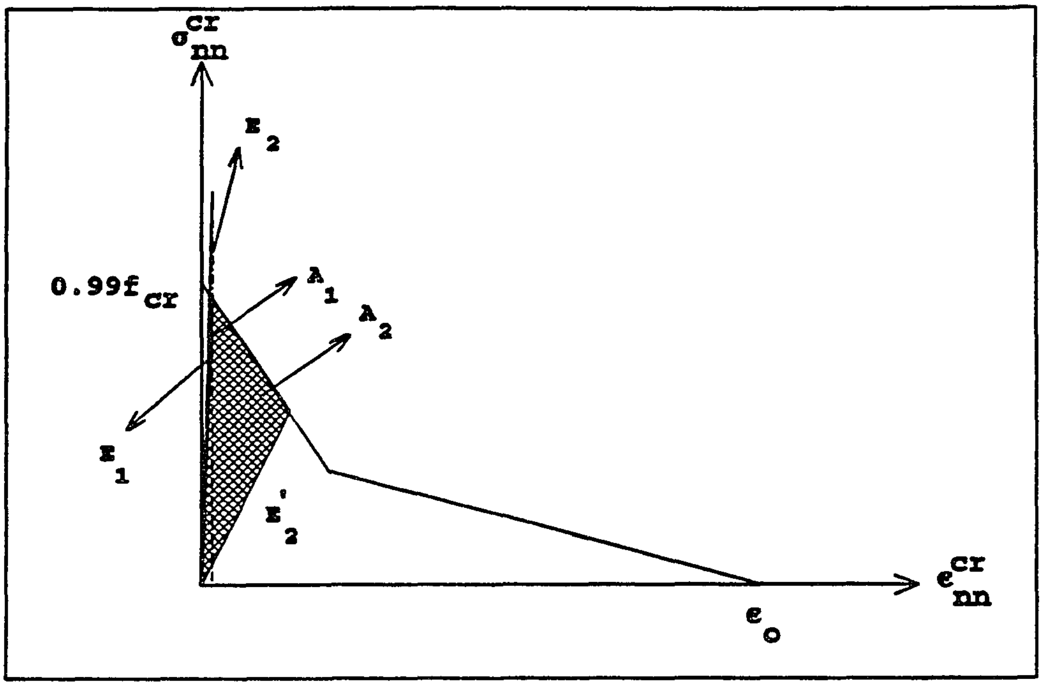


Figure 5.8: Detection and Correction of Overshooting at a Cracked Point.

relationship is satisfied.

$$A_1 \leq \text{tolerance} \; * A \tag{5.16}$$

If this criterion is violated then the new stiffness E_2 is employed and more iterations are required.

In the initial stages the tension stiffness is relatively high and the above criterion may be easily satisfied. However, the stress point may overshoots the cracking strength as shown in figure 5.10. In this case the convergence is rejected and a new stiffness E_3 should be employed. E_3 is obtained such that the area A_2 enclosed between E_1 and E_3 is given by:

$$A_2 = \text{tolerance} \; * A \tag{5.17}$$

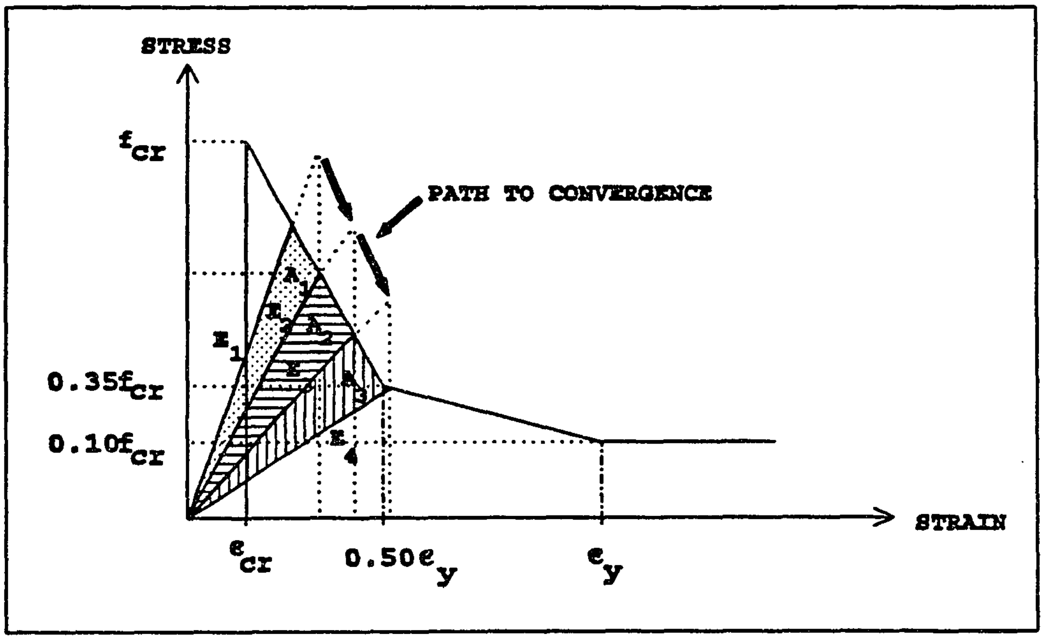


Figure 5.9: Convergence Path at a Tension-Stiffening Point.

5.6.1.4 Reinforcement Response

The global strain components obtained from displacements are transformed to the steel fibre orientation ϵ_n. If this strain component is less then the largest strain previously registered at this point the point is unloading and convergence is assumed. If the strain is found to be greater then the previously stored strain loading is assumed and a convergence criterion described here is enforced. Figure 5.11 shows the convergence procedure employed in the post-yield zone in steel. The stress in steel is obtained from the relationship $\sigma_s = E_1\epsilon_n$ where E_1 is the secant stiffness employed in the current iteration. The new stiffness E_2 is obtained from ϵ_n and the stress-strain diagram used for steel. If A is the total area under the stress-strain diagram for steel and A_1 is the area under the stress-strain diagram between E_1 and E_2 then convergence at the given sample point is assumed if the following criterion is satisfied:

$$A_1 \leq A * \text{ tolerance} \tag{5.18}$$

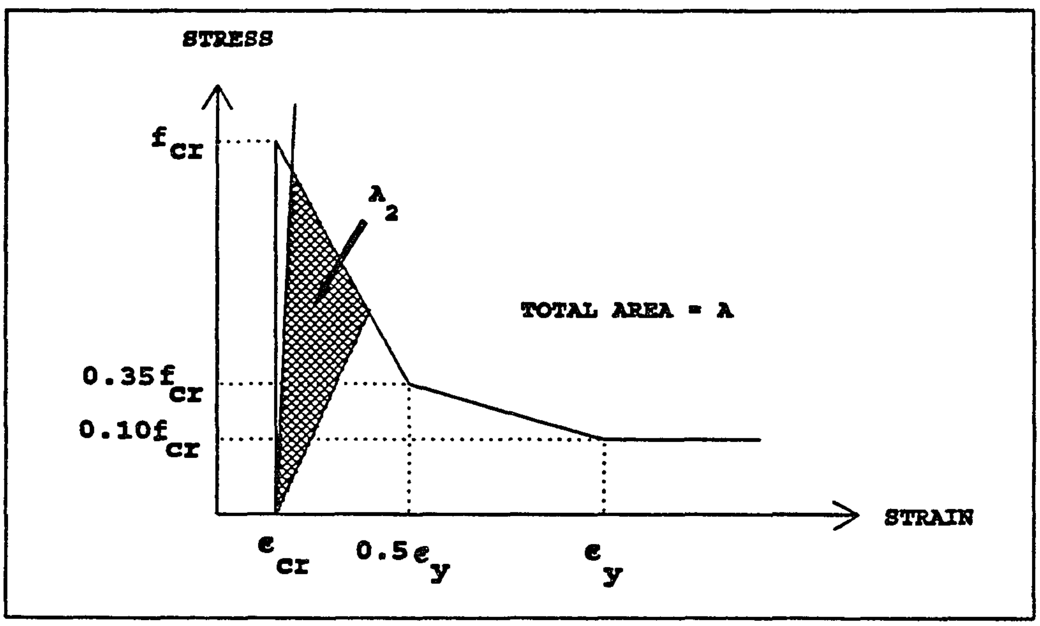


Figure 5.10: Detection and Correction for Overshooting at a Tension-Stiffening Point.

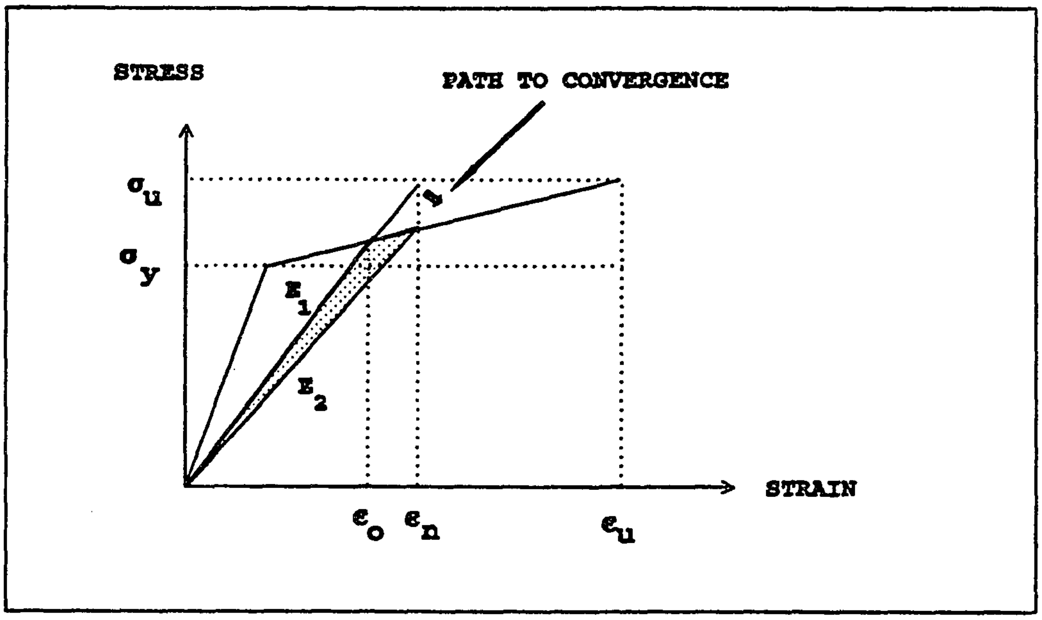


Figure 5.11: Convergence Path in Steel in the Post-Yielded Zone.

If the stress point σ_s overshoots the ultimate stress σ_u then this convergence is rejected (figure 5.12). The new stiffness E_2' is obtained such that the area A_2 between E_1 and E_2' meets the following requirement:

$$A_2 \leq A * \text{ tolerance} \tag{5.19}$$

5.6.2 Numerical Algorithm

The complete listing of the program along with sample input files are given in the appendix. A brief outline of the steps employed in the nonlinear algorithm are presented here.

1. For any given increment, the converged stiffness $\{K\}$ from the previous load increment and the current load vector [F] are used in equation 5.11 to solve for the displacements [U].

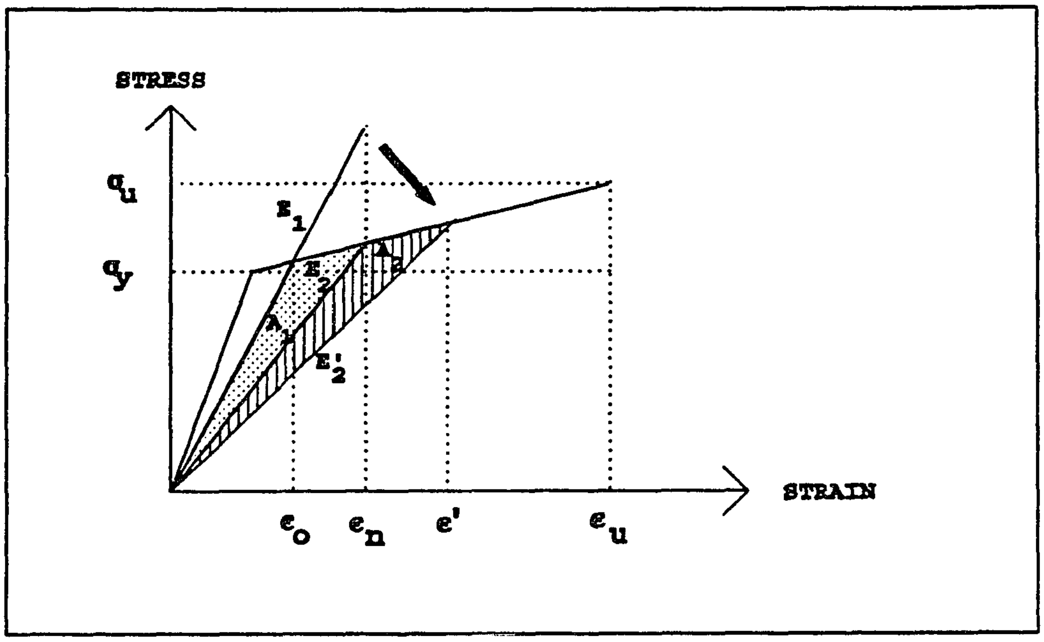


Figure 5.12: Detection and Correction for Overshooting in Steel.

2. From these displacements the total strains and total stresses are recovered at each sample point within each layer of each element.

3. Checks are made to determine the loading index at each point, within each rheological component based on the current strain and previously stored strain levels within each respective component. If this index indicates unloading, convergence is assumed.

4. If the loading index indicates the point is loading, the convergence criterion is employed to determine if the stiffness component is to be updated. If this is necessary then the convergence flag is set on and the stiffness matrix (equation 4.6) is reformulated at that point.

5. If each sample point within each layer of each element has been processed continue to step 6 else return to step 2.

6. If the convergence flag is on, update the stiffness matrix $\{K\}$ and return to step 1 provided the maximum number of iterations are not exceeded. If the iteration count exceeds the maximum allowable iterations (typically 50), go to step 8.

7. If convergence requirements have been satisfied, the results are printed. Return to step 1 with the next load increment. If all load increments have been applied stop the process.

8. If iterations have been exceeded the unconverged results are printed and the process is stopped.

5.6.3 Load Increment Size

To properly and efficiently capture the nonlinear behavior of reinforced concrete, it is necessary to keep the load increment size reasonably small. This provides a more accurate representation of the material behavior and requires fewer iterations to converge at each load step. Large load steps may force nonlinear behavior in some regions of the structure that may have remained elastic due to the softening in other parts of the structure. Load increments of 1 to 3% of the ultimate load have been used in this study.

Chapter 6

Numerical Studies and Discussion of Results

6.1 Introduction

To demonstrate the capabilities of the analytical model for reinforced concrete proposed in this study a number of RC test specimen under the action of various loads have been examined. The details of these studies are reported in this chapter.

First, the material model parameters required to establish the shape of the tension-stiffening spring, described in section 4.3, have been evaluated by analyzing a number of test specimen subjected to uniaxial tension. Panel elements under the action of uniform membrane stress fields have been examined to highlight the influence of the different components of the analytical model in capturing the dominant response behavior.

Wall elements under the action of vertical and horizontal in-plane loads, applied proportionally and non-proportionally, have then been analyzed. Beam and slab components have also been analyzed to evaluate the model capabilities under the action of flexural and torsional loads. The load-deformation response, cracking patterns, stress and strain contours in concrete and steel have been examined to determine the dominant response characteristics and modes of failure of these specimen. Appropriate comparisons with results from other investigations and the ACI code provisions have also been included.

6.2 RC Elements Subjected to In-plane Loads

6.2.1 Membrane Loading Without Stress Gradients

6.2.1.1 Uniaxial Loadings

Uniaxial tension test specimen were selected for this study primarily to establish the post cracking stress-strain response curves of reinforced concrete employed in the analysis. The criterion used for selecting test specimen included the shape of RC element, the percentage of reinforcement, the orientation of reinforcement and the material properties of concrete and steel. To consider the influence of this broad range of parameters the examples were selected from Rizkalla and Hwang (1984), Shima et al. (1987), and Bhide and Collins (1987).

The parameters defining the shape of the tension-stiffening curve shown in figure 4.4, in section 4.3.1, were obtained by calibrating them to the uniaxial test results discussed here. The values of α, β, γ and δ (figure 4.4) obtained were $0.5\epsilon_y$, ϵ_y, $0.35f_{cr}$, and $0.10f_{cr}$, respectively, where ϵ_y is the strain in steel at yield and f_{cr} is the cracking stress in concrete.

The dimensions and material properties of the test specimens is presented in table 6.1. Only a single shell element is used to model the specimen since the stress field is uniform. The cross-section of the RC specimen were modelled using the implicit layering procedure described in section 5.5.3. A single layer is used to model the concrete, while steel in each different direction is treated as an individual layer with the appropriate layer thickness. A 3x3x1 Gaussian numerical integration procedure has been employed to compute the stiffnesses of both concrete and steel. Figure 6.1 shows the tensile stress-strain behavior of a reinforced concrete bar Shima #3 (specimen number), tested by Shima et al. (1987). The load deformation response of this bar is shown in figure 6.2. The analytical results obtained in the present study are also presented in these figures and are labeled as FEM1. Both the axial load-strain as well as the concrete tensile stress-strain response characteristics appear to be well represented by the analytical prediction up to ultimate. The oscillations in the tensile stress-strain response in the initial stages of the analysis is due to excessive reduction of the concrete stiffness and corrects itself with additional loading. The mode of failure was yielding of the steel. The ultimate load as reported in table 6.2 compares well with the experimental results.

Table 6.1: Dimensions and Material Properties of Uniaxial Tension Specimen

Speci. Label	Dimensions (mm)	Concrete			Reinforcement		
		f'_c (MPa)	ϵ'_c $\frac{mm}{mm}$	f'_t (MPa)	$\rho^{tran.}$ (%)	$f_y^{tran.}$ (MPa)	Orient. (deg.)
Shima#3	2700x150x200	25.7	0.002	0.8	1.	610	0.0
Shima#5	2700x250x250	25	0.002	1.6	1.47	820	0.0
Rizkalla#2	762x305x178	34.5	0.002	2.9	1.47	453	0.0
PB13	890x890x70	23.4	0.002	1.48	1.085	403	0.0
Note: Initial Stiffness in Concrete is $\frac{2f'_c}{\epsilon_c}$							

Table 6.2: Results of Uniaxial Tension Tests on RC Specimen

Specimen	Finite Element Analysis		Experiment		% Error
	Ultimate Load (MPa)	Failure Mode	Ultimate Load (MPa)	Failure Mode	
Shima #3	6.10	Yielding	6.01	Yielding	1.4
Shima #5	5.08	Yielding	4.82	Yielding	5.4
Rizkalla #2	7.1	Yielding	6.55	Yielding	8.4
PB13	4.76	Yielding	4.70	Yielding	1.2

Specimen Shima #5 (specimen number), tested by Shima et al.(1987), has also been analysed in this study. This specimen has a higher percentage of steel of 2%, compared to 1% in specimen Shima #3. The ultimate load obtained from the analyses and the experiment is reported in table 6.2. As expected, the increase in reinforcement percentage is reflected in a higher load carrying capacity. The tension-stiffening formulation employed in the analysis is not explicitly dependent on the percentage of reinforcement present in the specimen. However, the analytical

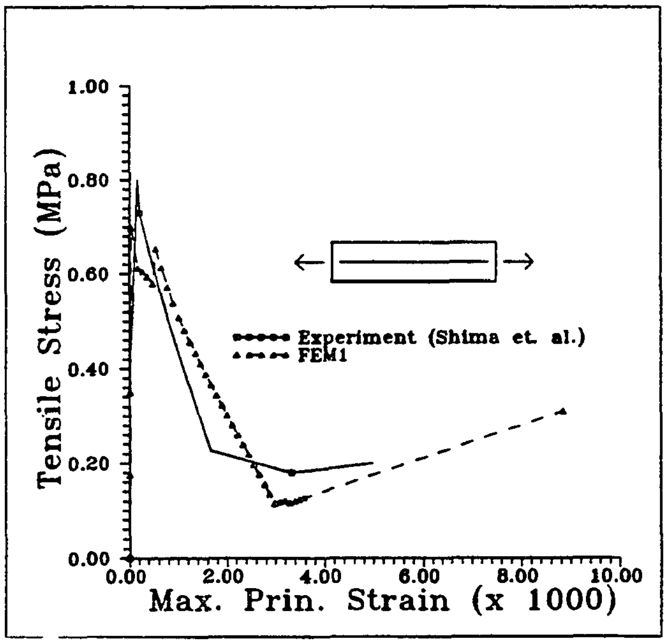


Figure 6.1: Tensile Stress-Strain Response, Specimen Shima #3.

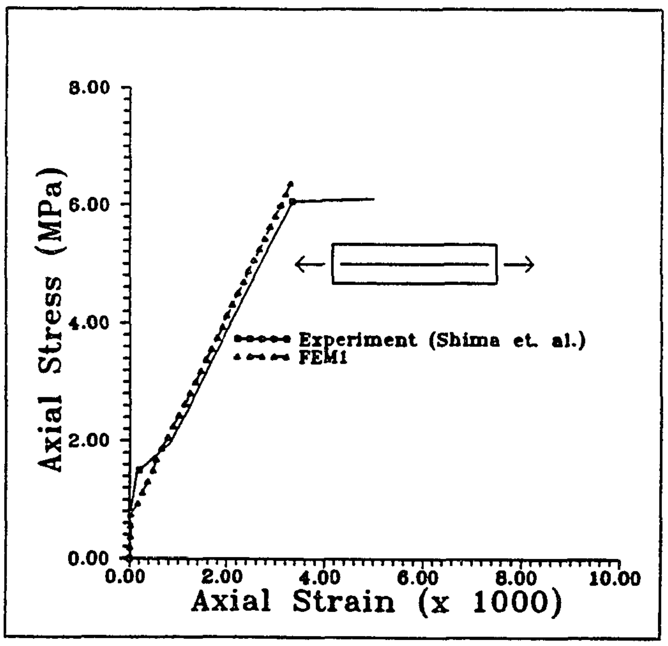


Figure 6.2: Axial Load-Deformation Response, Specimen Shima #3.

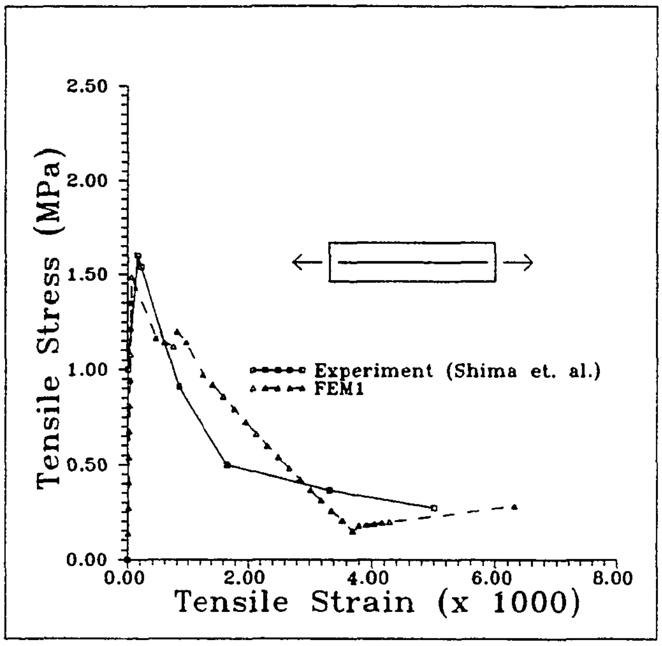


Figure 6.3: Tensile Stress-Strain Response, Specimen Shima #5.

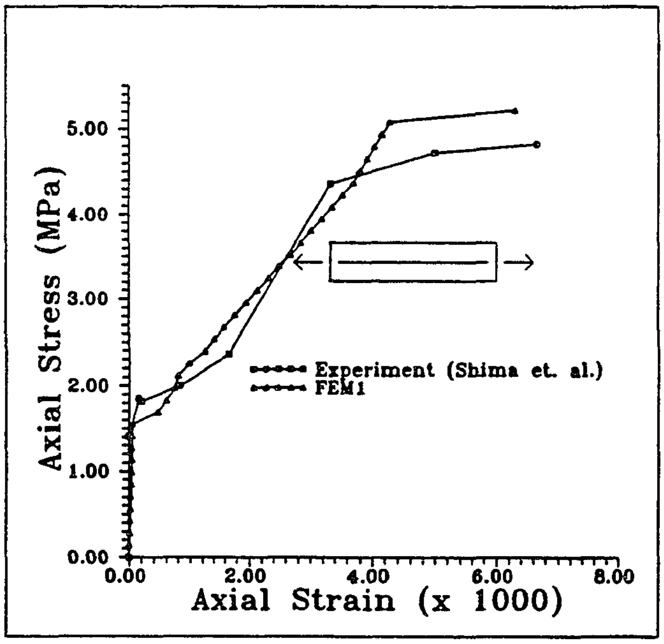


Figure 6.4: Axial Load-Deformation Response, Specimen Shima #5.

prediction of the tensile stress-strain response and the the load deformation response is very close to the experimental result (figure 6.3 and 6.4). Thus, the use of a smeared representation of the reinforcement and tension stiffening for a single bar appears to be quite effective in simulating the load deformation responses.

While the tests conducted by Shima et al. (1987) were on RC bars, those conducted by Rizkalla and Hwang (1984) and Bhide and Collins (1986) were done using RC panel elements. Figure 6.5 shows the tensile stress-strain response predictions for specimen Rizkalla #2 along with the experimental results from Rizkalla and Hwang (1984). The analytical predictions closely follow the experimental results up to failure. The axial load-deformation response for this specimen is shown in figure 6.6. The ultimate load is reported in table 6.2. Specimen PB13, tested by Bhide and Collins (1987), has also been analyzed in this study and the ultimate load is given in table 6.2. The tensile stress-strain response and the load deformation response for specimen PB13 is shown in figure 6.7 and 6.8 respectively.

The analytical model does not have any size dependent parameters built into it explicitly. However, the tensile stress-strain response obtained from the analytical model is quite satisfactory when compared with the experimental results. This would indicate that the the size and shape of the specimen do not appear to have influenced the tensile stress-strain response and that employing a smeared tension-stiffening layer to represent the transfer of stresses through bond is reasonable.

6.2.1.2 General In-plane Loading

In the preceding section, the capabilities of the analytical model in predicting the response of uniaxially loaded tensile test specimen has been reported. In this section, its ability to predict the response of test specimen under uniform general membrane stress fields is examined. Unless otherwise stated a threshold angle of 15° has been used, for the initiation of a new crack with respect to previously existing cracks, in these analyses. A single shell element is used to describe the specimen, since only uniform stress fields are present in these specimen. A 3x3x1 Gaussian numerical integration procedure has been employed for computing the stiffnesses of

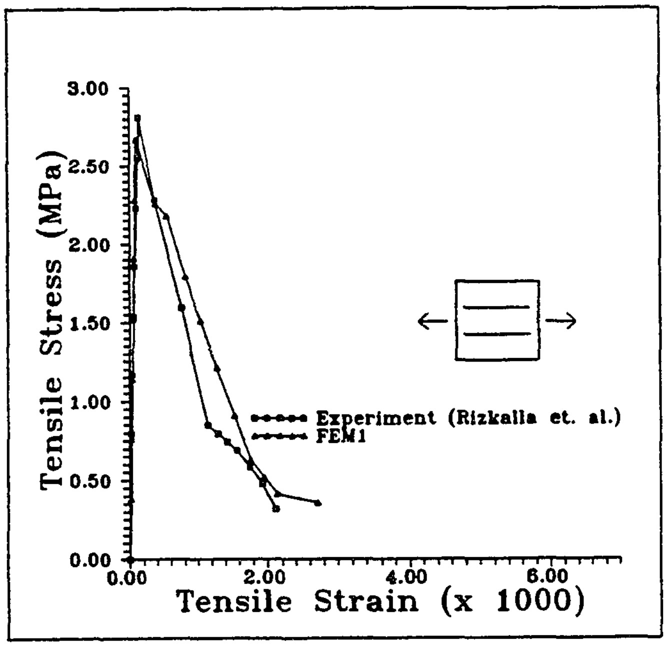


Figure 6.5: Tensile Stress-Strain Response, Specimen Rizkalla #2.

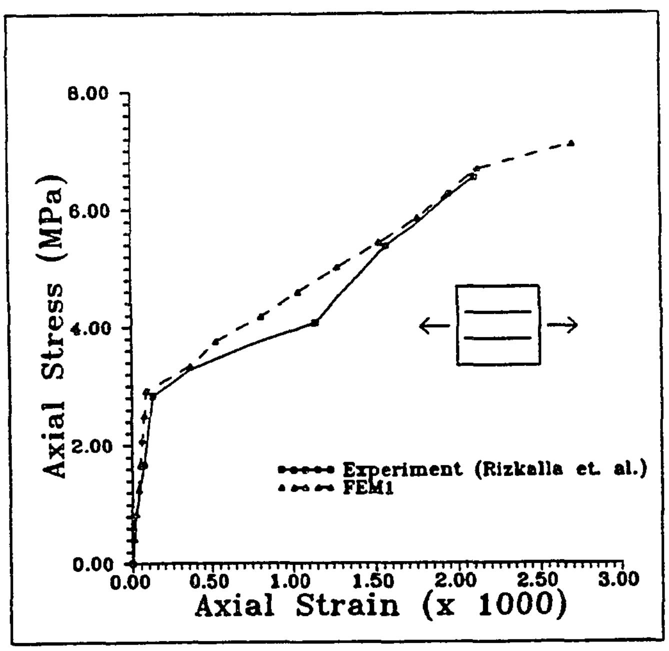


Figure 6.6: Axial Load-Deformation Response, Specimen Rizkalla #2.

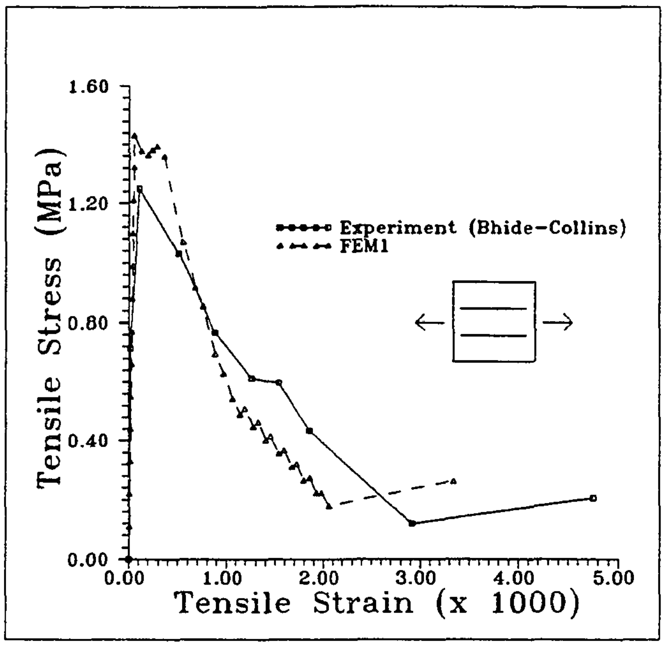


Figure 6.7: Tensile Stress-Strain Response, Specimen PB13.

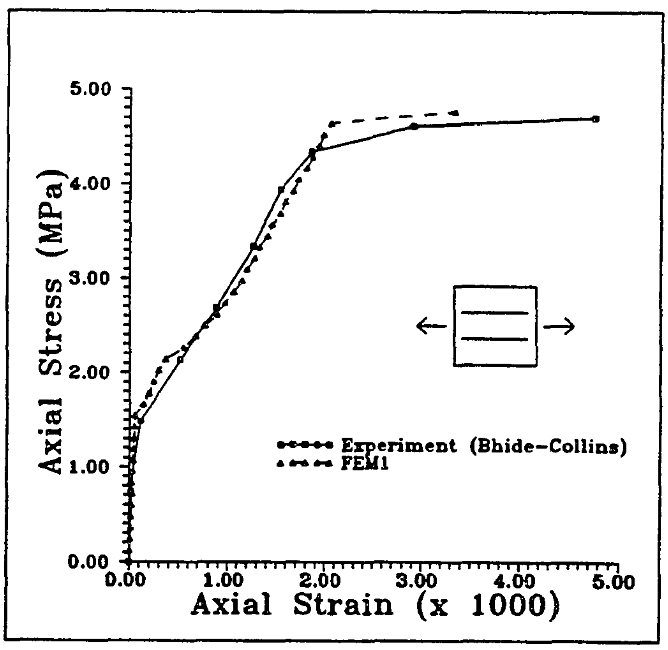


Figure 6.8: Axial Load-Deformation Response, Specimen PB13.

the steel and concrete layers. The cross-section of the RC specimen were modelled using the implicit layering procedure described in section 5.5.3. A single layer is used to model the concrete, while each set of steel bars in a direction is treated as a separate layer with the appropriate thickness.

Panels S1, S3, S5, S6 and S7

To study the influence of the orientation of the reinforcement and anisotropic arrangement of the reinforcement on the accuracy of prediction of the analytical model, a series of panel elements tested by Dyngland (1989) were selected in this study. This series of specimen consisted of three isotropically reinforced specimen (S1, S3 and S6) and two with anisotropic arrangement (S5, S7) of steel. The material properties, type of loading and geometry of the test specimen is given in tables 6.3 and 6.4 These specimen were all loaded in uni-directional tension. Figures 6.9, 6.10 and 6.12 show the applied load deformation response of isotropically reinforced specimen S1, S3 and S6 respectively. The analytical predictions of the deformational response appear to closely follow the experimental results in each case indicating the use of tension stiffening in direction(s) skewed to the loading direction (also the direction of initial cracks) is reasonable.

Table 6.3: Loading Procedures for Tests on Membrane Elements

Specimen	Loading
S1	Uniaxial Tension
S3	Uniaxial Tension
S5	Uniaxial Tension
S6	Uniaxial Tension
S7	Uniaxial Tension
PB18	Pure-Shear
PB19	Tension:Shear :: 1.01:1
PB20	Tension:Shear :: 2.04:1
PB21	Tension:Shear :: 3.08:1
PB22	Tension:Shear :: 6.09:1

Analytical predictions of the response of specimen S5, shown in figure 6.11, indicates a good match with the experimental results of Dyngland (1989). As expected,

Table 6.4: Dimensions and Material Properties of Membrane Elements

Speci. Model	Dimensions (mm)	Concrete f'_c (MPa)	ϵ'_c $\frac{mm}{mm}$	f'_t (MPa)	Reinforcement $\rho^{trans.}$ (%)	$f_y^{trans.}$ (MPa)	$\rho^{long.}$ (%)	$f_y^{long.}$ (MPa)
S1[1]	630x630x100	23.1	0.002	2.1	1.12	403	-	-
S3[2]	630x630x100	21.8	0.002	2.1	1.12	403	1.12	403
S5[2]	630x630x100	22.9	0.002	2.1	0.59	403	1.12	403
S6[3]	630x630x100	24.3	0.002	2.1	1.12	403	1.12	403
S7[3]	630x630x100	23.1	0.002	2.1	1.12	403	0.37	403
PB18[1]	890x890x70	25.3	0.002	1.62	2.195	402	-	-
PB19[1]	890x890x70	20.	0.002	1.65	2.195	402	-	-
PB20[1]	890x890x70	21.7	0.002	1.75	2.195	424	-	-
PB21[1]	890x890x70	21.8	0.002	1.95	2.195	402	-	-
PB22[1]	890x890x70	17.6	0.002	1.75	2.195	433	-	-

1:Reinforcement mesh at 0^o to loading direction
2:Reinforcement mesh at 45^o to loading direction
3:Reinforcement mesh at 18.4^o to loading direction
Concrete initial Stiffness $\frac{2f'_c}{\epsilon_c}$

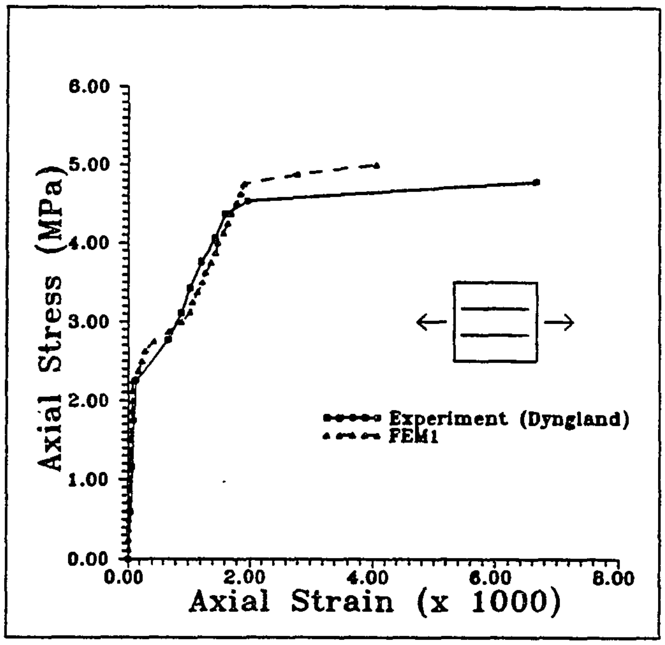


Figure 6.9: Axial Stress-Strain Response, Specimen S1.

the rotation of the principal directions towards the weakly reinforced direction is observed in the analysis. This indicates that the development of the shear stresses on the crack surface, that are responsible for the rotation of the principal directions, are well represented by the crack interface shear model employed in this study.

Figure 6.13 shows the analytical prediction for specimen S7 together with the experimental result. While the ultimate load predicted for this specimen is close to the experimental one, the deformational response predicted is more flexible than the experimental results after initial cracking due to 'global' yielding of the reinforcement in the weaker direction. One possible reason for the stiffer response observed in the experiment, is the resistance offered by the welds connecting the orthogonally reinforced bars (forming a mesh) to straining in the weak direction, which is not accounted for in the analysis. Thus, while the weaker direction yielded in the experiment the strains registered in that direction were reduced compared with those obtained in the analysis.

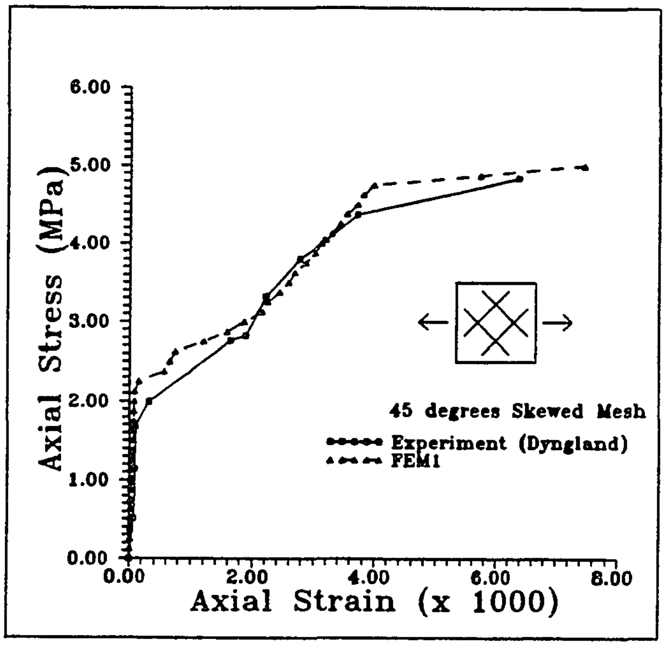


Figure 6.10: Axial Stress-Strain Response, Specimen S3

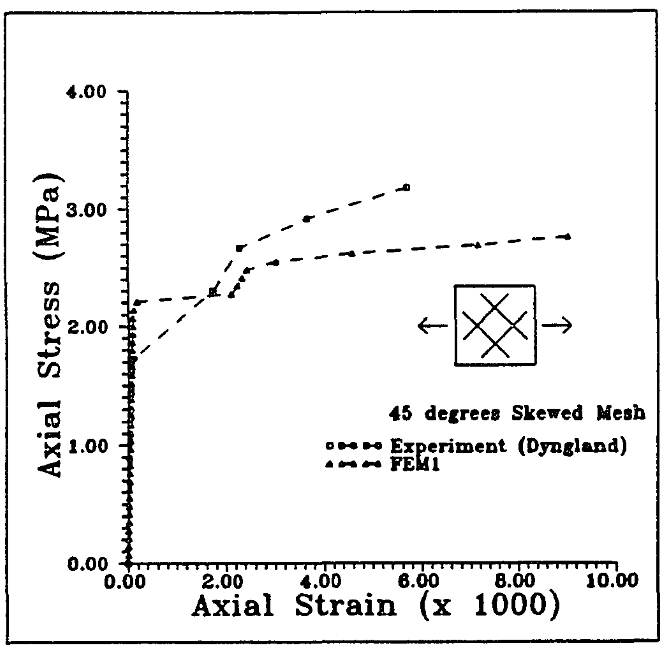


Figure 6.11: Axial Stress-Strain Response, Specimen S5

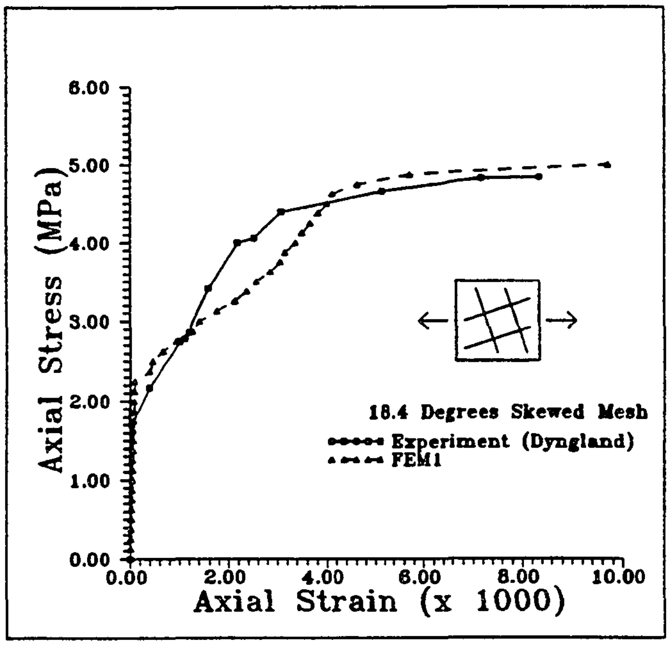


Figure 6.12: Axial Stress-Strain Response, Specimen S6

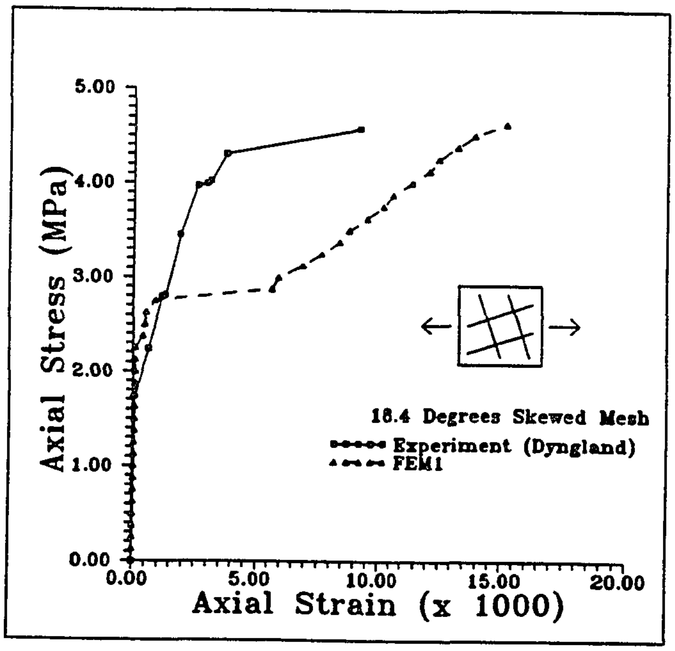


Figure 6.13: Axial Stress-Strain Response, Specimen S7

Table 6.5: Results of Tests on Membrane Elements

Specimen	Finite Element Analysis		Experiment		% Error
	Ultimate Load (MPa)	Failure Mode	Ultimate Load (MPa)	Failure Mode	
S1	5.0	Yielding	4.78	Yielding	4.6
S3	5.0	Yielding	4.84	Yielding	3.3
S5	2.76	Yielding	3.18	Yielding	13.2
S6	5.0	Yielding	4.84	Yielding	3.3
S7	4.63	Yielding	4.57	Yielding	1.3
PB18	1.87[1] (2.23)[2]	Sliding	1.65	Sliding	13.3
PB19	1.58[1] (2.05)[2]	Sliding	1.23	Sliding	28.4[1]
PB20	1.26	Sliding	1.41	Sliding	10.6
PB21	1.37	Sliding	1.38	Sliding	0.7
PB22	0.97	Sliding	1.01	Sliding	3.9

1 : Threshold Angle 10^o
2 : Threshold Angle 15^o

Panel PB21: Shear and Tension

Panel PB21, tested by Bhide and Collins (1987) is a uniaxially reinforced panel subjected to simultaneous shear and tension. The material properties, geometry and loading information are given in table 6.3. The initial crack formed (defined by the crack normal direction) at an 18^o skew with the reinforcing direction. Due to the lack of reinforcement in one direction, a significant shearing stress is generated on the crack plane to equilibrate the external forces in the unreinforced direction. This causes the principal loading direction to rotate rapidly towards the unreinforced (weak) direction, leading eventually to further cracking in a new direction and failure. In the analysis, these secondary cracks form at an inclination of nearly 34^o with respect to the reinforcement. The shear stress-strain response, shown in figure 6.14 indicates that the analytical prediction compares well with the experimental results. The tensile stress-strain response is shown in figure 6.18. The stress-strain response in the reinforcing direction, shown in figure 6.15, appears to be well represented by the analysis. The stress-strain response in the unreinforced direction (figure 6.16) shows that the analytical response is stiff compared to that observed in the experiment. While the initial response is controlled by the sliding behavior on the crack plane the formation of secondary cracks and the large straining in the unreinforced direction is the cause of 'failure'. This indicates that beyond a certain skew between the crack direction and the reinforcement the crack response is like that of a plain concrete specimen and the influence of the reinforcement and the tension-stiffening on the load carrying capacity is negligible. The load-maximum principal strain response is shown in figure 6.17. The variation in the direction of the principal stress and strain direction with applied loads is shown in figure 6.19 and 6.20 respectively. The analytical predictions of the rotations in the principal stress direction compares reasonably well with the experimental results. Thus, the crack interface shear stresses modelled by the aggregate interlock model, responsible for rotation of the principal stress directions, appears to be well represented. The severe rotation of the principal strain direction is not well simulated by the analysis. The reason for this is the delay in the formation of secondary cracks imposed by the minimum threshold angle (15°) requirement as discussed in section 5.3 (simulation of cracks). The experimental results indicate a rapid rotation of the principal strain direction after initial cracking and formation of secondary cracks at a load of 1.07MPa. The formation of secondary cracks is initiated only at a load of 1.17MPa in the analysis. The ultimate load prediction and the experimentally obtained values, given in table 6.5, compare reasonably.

Panel PB18: Pure Shear

Panel PB18 is a uniaxially reinforced panel subjected to pure shear load. The material properties, geometry and loading information are given in table 6.3. The initial cracks formation is at 45° to the reinforcing direction. Due to the lack of reinforcement in one direction, a significant shearing stress is generated on the crack plane to balance the external forces in the unreinforced direction. This causes the principal loading direction to rotate rapidly towards the unreinforced direction, leading to further cracking and failure almost immediately after the first cracks are formed. This test conducted by Bhide and Collins (1987) is a case of extreme anisotropy. The shear stress-strain response is shown in figure 6.21. The analytical response is

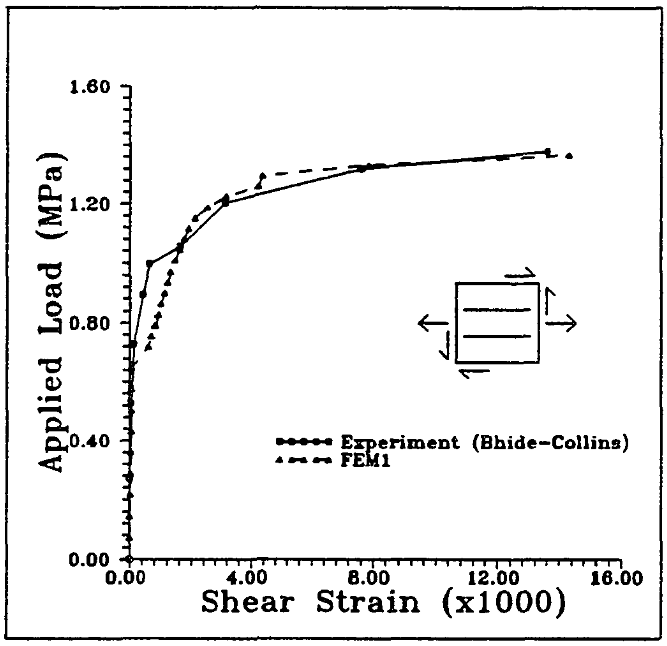


Figure 6.14: Load vs. Strain Response, Specimen PB21

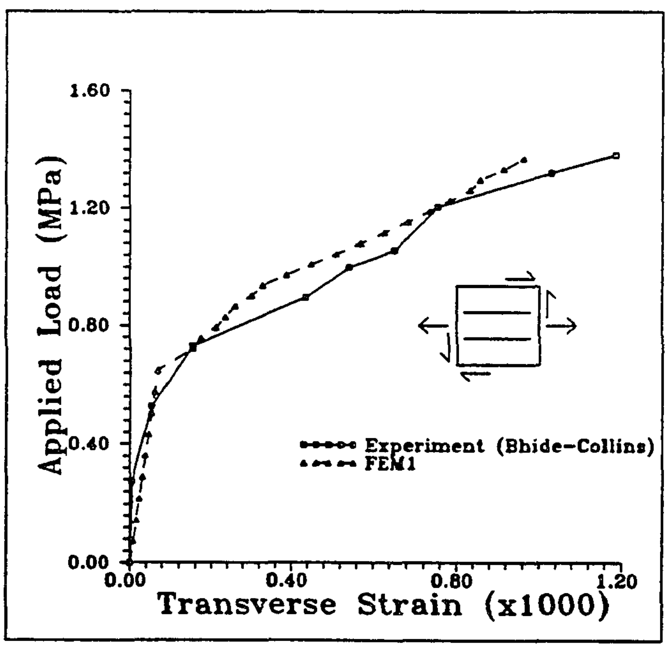


Figure 6.15: Load vs. Transverse Strain Response, Specimen PB21

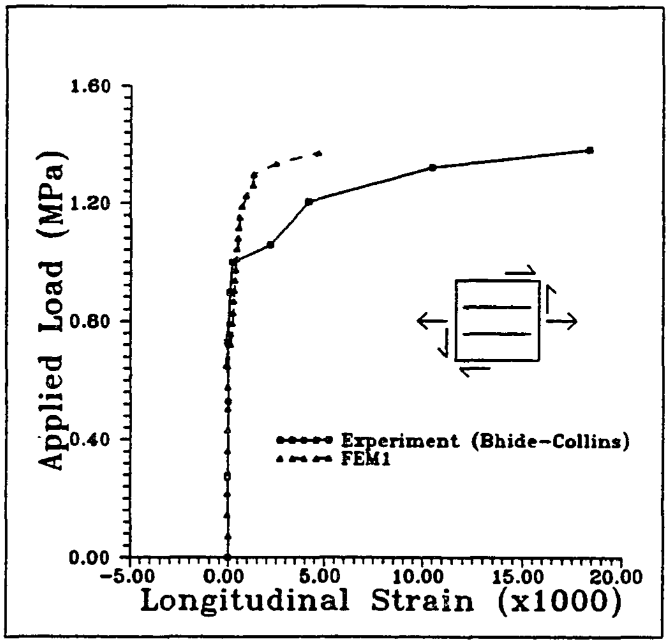


Figure 6.16: Load vs. Longitudinal Strain Response, Specimen PB21

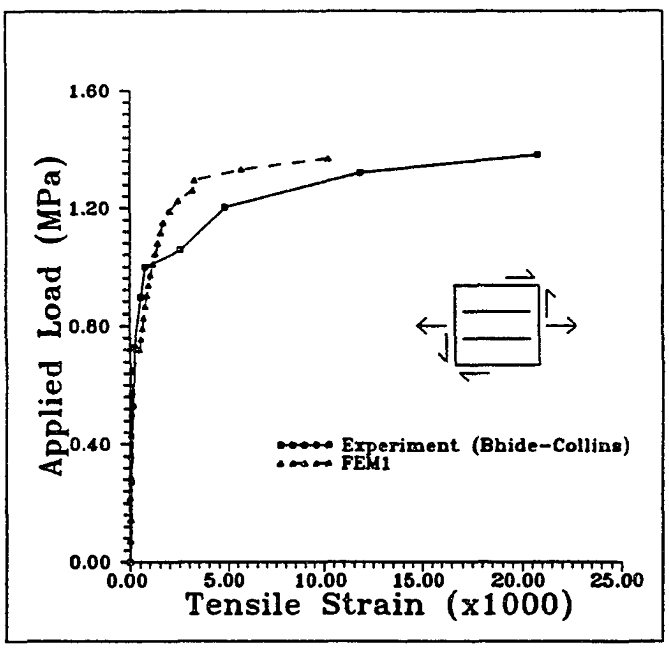


Figure 6.17: Load vs. Maximum Principal Strain Response, Specimen PB21

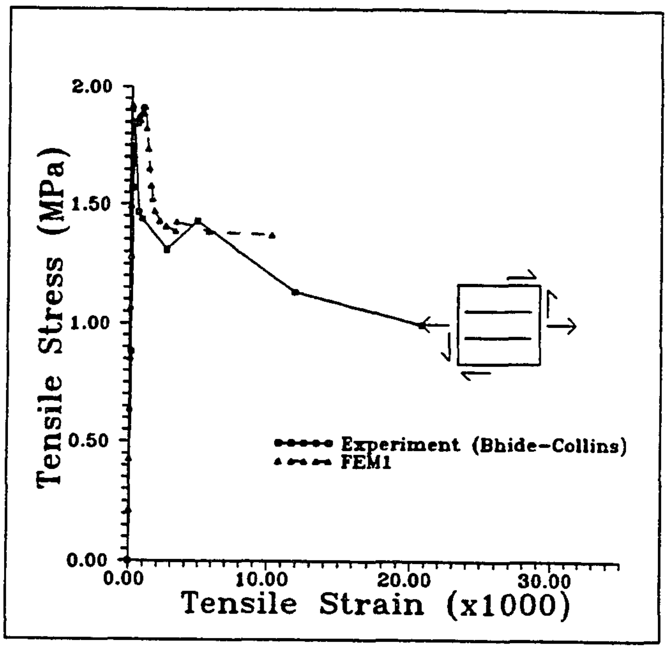


Figure 6.18: Tensile Stress-Strain Response, Specimen PB21

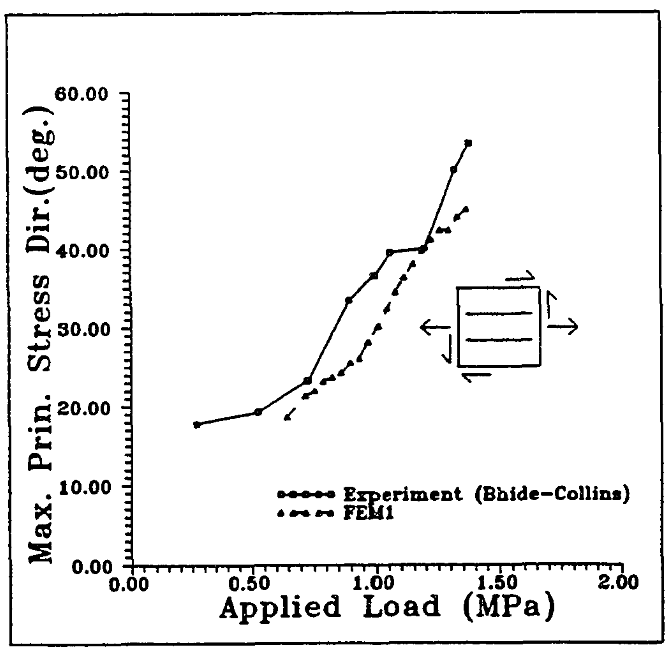


Figure 6.19: Principal Stress Direction vs. Axial Load, Specimen PB21

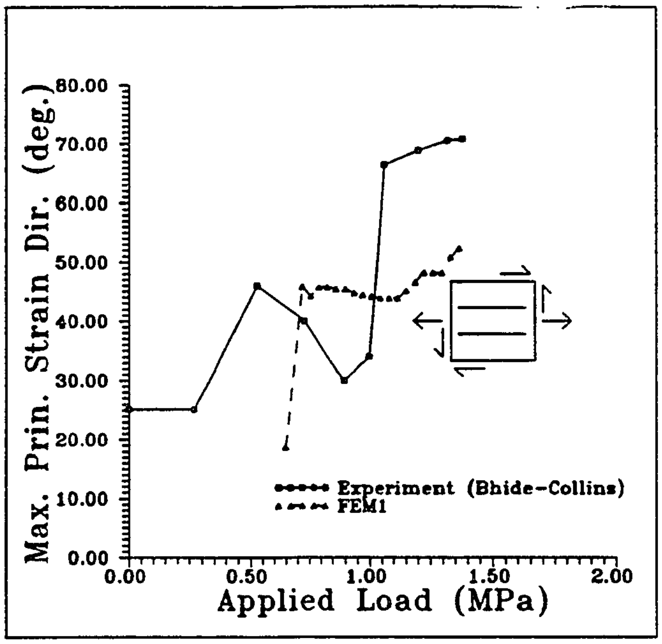


Figure 6.20: Principal Strain Direction vs. Axial Load, Specimen PB21

shown for two allowable threshold angles established for crack formation. When the threshold angle is 15° the secondary crack formation is delayed significantly, resulting in a very stiff load deformation response. When the threshold angle is reduced to 10° the ultimate load is reduced and the results are closer to that observed in the experiments. The tensile stress-strain response is shown in figure 6.23. The stress-strain response in the reinforcing direction, shown in figure 6.22, appears to be well represented by either analysis. This implies that beyond a certain skew between the cracks formed and the reinforcement the crack response is like that of a plain concrete specimen and the influence of the reinforcement on their response is negligible. The stress-strain response in the uncracked direction (figure 6.25) shows that the analytical response is stiff compared to that of the experiment. This indicates that while the initial response is controlled by the sliding behavior on the crack plane, the formation of secondary cracks and the large straining is the cause of 'failure'. The load-maximum principal strain response is shown in figure 6.24.

The load-principal stress and strain direction responses are shown in figure 6.26

and 6.27 respectively. Results of other tests conducted in this series by Bhide and Collins (1987), specimen PB19 PB20 and PB22, are shown in table 6.5 along with the ultimate load prediction obtained in this study.

Panel PV19: Pure Shear

RC panel PV19 is an anisotropically reinforced panel element subjected to pure shear tested by Vecchio and Collins (1982). The reinforcement in the transverse direction is lower (0.78%) compared to the longitudinal direction (1.78%). After cracking, this anisotropy causes the principal stress and strain directions to rotate toward the weaker direction as seen in figure 6.33 and figure 6.34 respectively. The increased straining along the weaker direction causes the transverse steel direction to yield as observed in figure 6.30. The longitudinal strain component increases very slowly and remains in the elastic range at failure (figure 6.29). The tensile stress-strain response from the tests, shown in figure 6.31, clearly shows that some tensile stresses are sustained by the concrete, even after yielding of the transverse steel. This is also reproduced in the analysis. After cracking, a large amount of the load transfer capacity is generated in the weaker direction through the crack interface shear stresses. However, with the rapid increase in loads, this interface shear transfer capacity is exhausted and a shear sliding failure is predicted by the analysis FEM1 (figure 6.28). The compressive stress-strain behavior obtained from analysis FEM1 as seen in figure 6.32 clearly indicates that the capacity of the compressive strut is not exhausted. The analytical model FEM2 predicts crushing of the compressive strut due to the strain-based reduction of the uniaxial compressive strength (nearly 40% reduction). The ultimate load predicted by FEM1 is 4.0 MPa while FEM2 predicted failure at 3.55 MPa. The experimental failure load of 3.9MPa (table 6.8). More significantly the failure mode observed in the experiment was of failure along the shear plane as predicted by FEM1. Premature crushing observed in FEM2, was caused by excessive softening and strength reduction of the compressive strut.

Panel PV27: Pure Shear

RC panel element PV27 is an isotropically reinforced specimen ($\rho = 1.75\%$) subjected to pure shear.The shear stress-strain behavior is shown in figure 6.35.It is

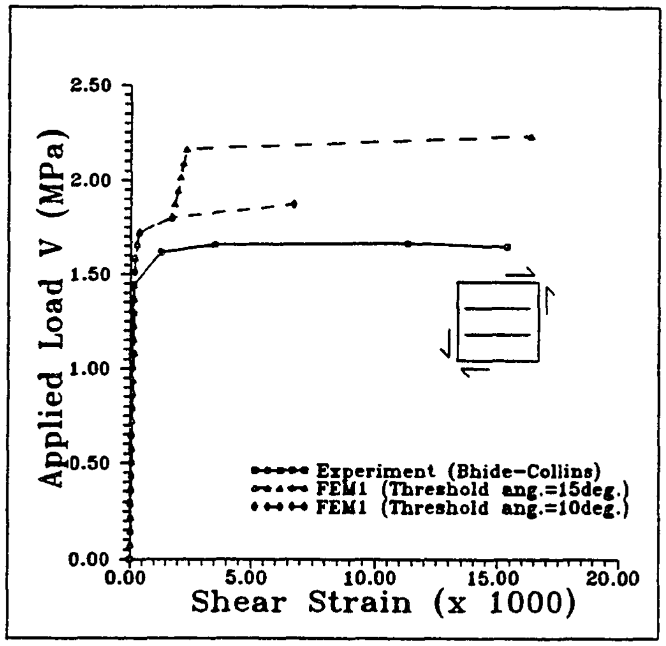


Figure 6.21: Load vs. Strain Response, Specimen PB18

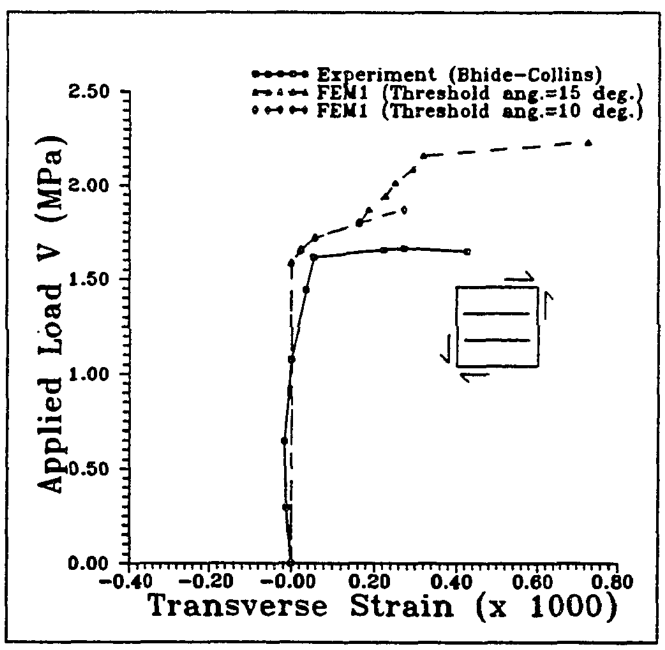


Figure 6.22: Load vs. Transverse Strain Response, Specimen PB18

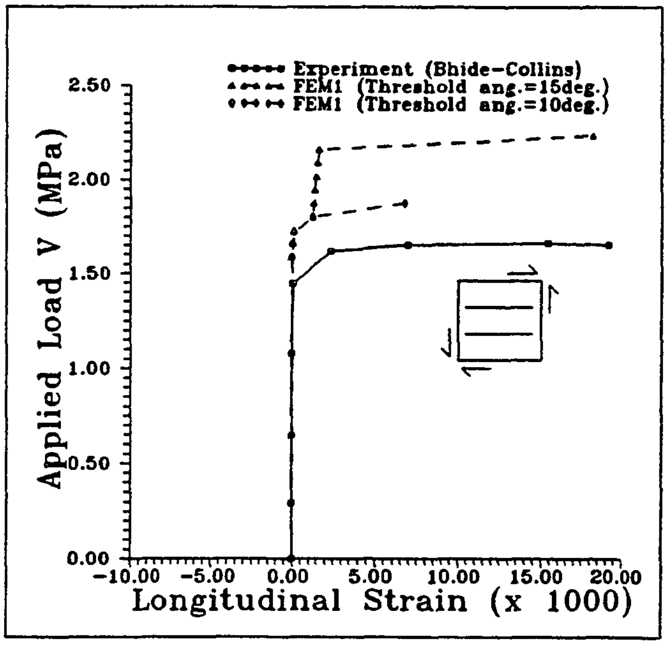


Figure 6.23: Load vs. Longitudinal Strain Response, Specimen PB18

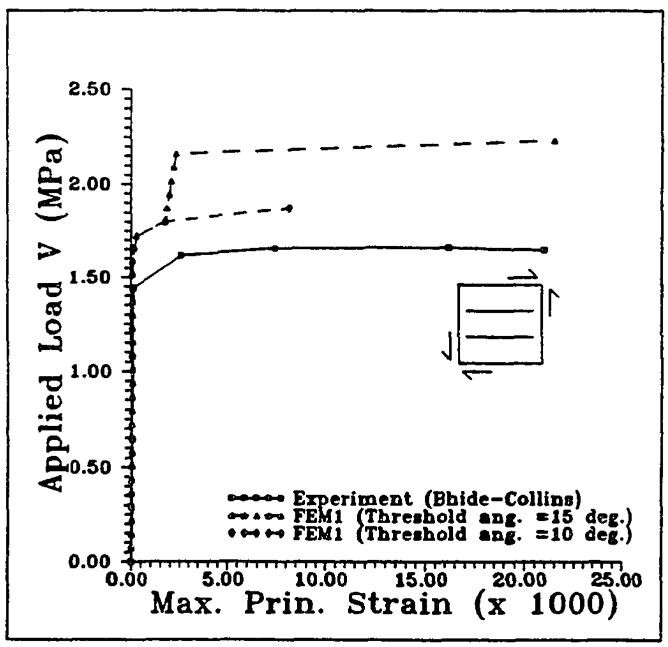


Figure 6.24: Load vs. Maximum Principal Strain Response, Specimen PB18

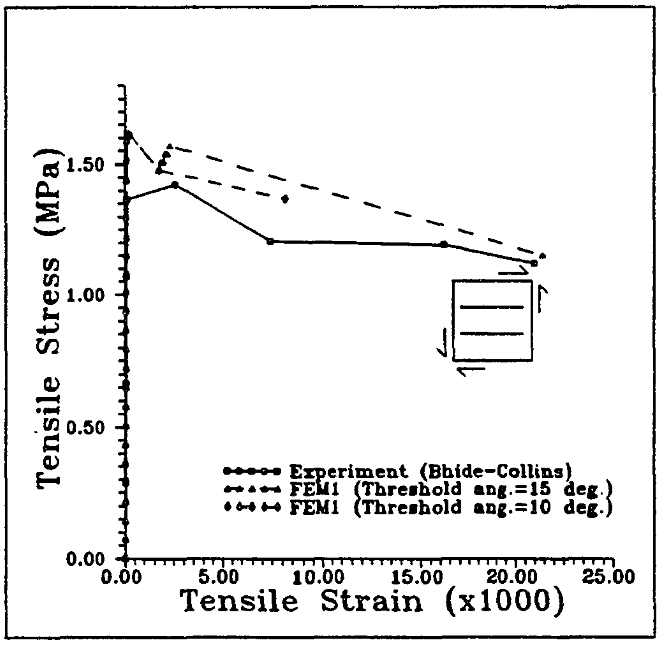


Figure 6.25: Tensile Stress-Strain Response, Specimen PB18

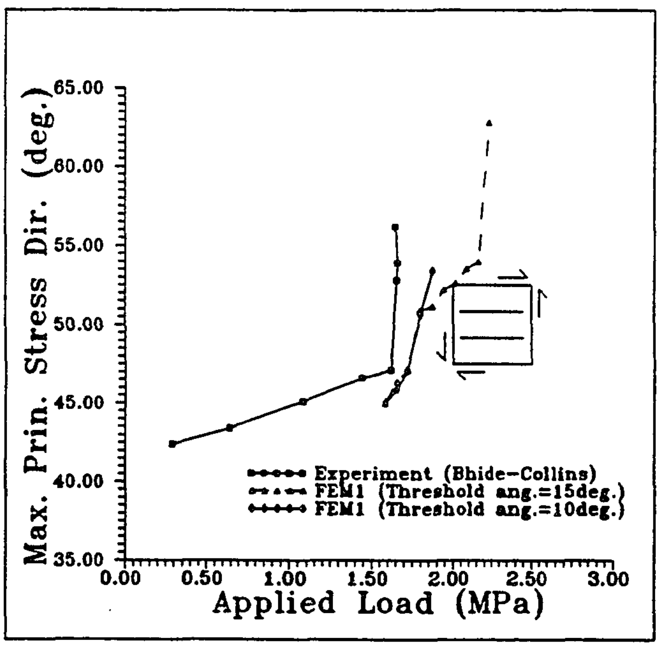


Figure 6.26: Principal Stress Direction vs. Axial Load, Specimen PB18

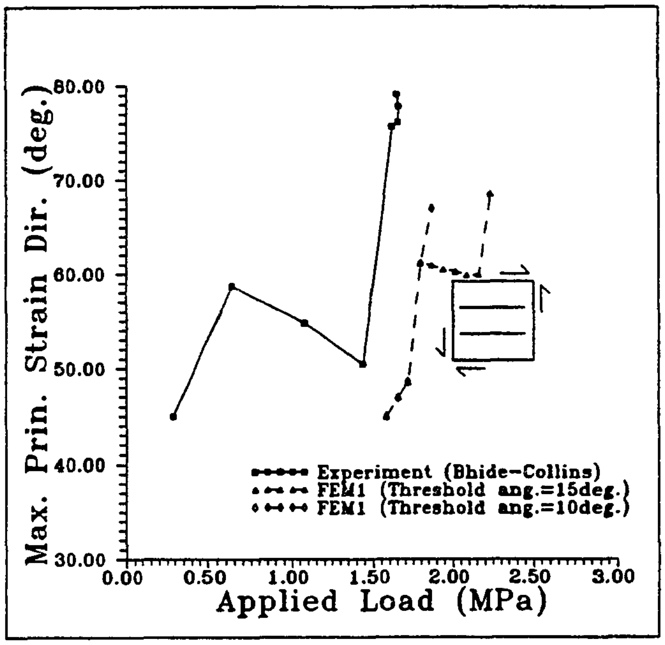


Figure 6.27: Principal Strain Direction vs. Axial Load, Specimen PB18

Table 6.6: Loading Procedures For Tests on Membrane Elements

Specimen	Loading
PV19	pure-shear
PV27	pure-shear
PV29	Non prop. shear-comp.
CS6	Non prop. tension-comp.

Table 6.7: Dimensions and Material Properties of Membrane Elements

Speci. Model	Dimensions (mm)	Concrete			Reinforcement			
		f'_c (MPa)	ϵ'_c $\frac{mm}{mm}$	f'_t (MPa)	$\rho^{trans.}$ (%)	$f_y^{trans.}$ (MPa)	$\rho^{long.}$ (%)	$f_y^{long.}$ (MPa)
PV19	890x890x70	19.0	0.002	1.9	0.713	299	1.785	458
PV27	890x890x70	20.5	0.002	2.6	1.785	442	1.785	442
PV29	890x890x70	21.7	0.002	2.1	.885	324	1.785	441
CS6	630x630x100	20.96	0.002	1.5	1.12	500	1.12	403
Note: Concrete Initial Stiffness $\frac{2f'_c}{\epsilon_c}$								

Table 6.8: Results of Tests on Membrane Elements

Specimen	Finite Element Analysis		Experiment		% Error
	Ultimate Load (MPa)	Failure Mode	Ultimate Load (MPa)	Failure Mode	
PV19	$4.04^1(3.48)^2$	Sliding	3.95	Sliding	2.2
PV27	$7.62^1(6.19)^2$	Crushing	6.25	Crushing	20.3
PV29	$7.01^1(7.14)^2$	Crushing	5.57	Crushing	20.5
CS6	$19.93^1(19.2)^2$	Crushing	18.89	Crushing	5.5^1
1: FEM1 Stress-Based Compressive Softening					
2: FEM2 Strain-Based Compressive Softening					

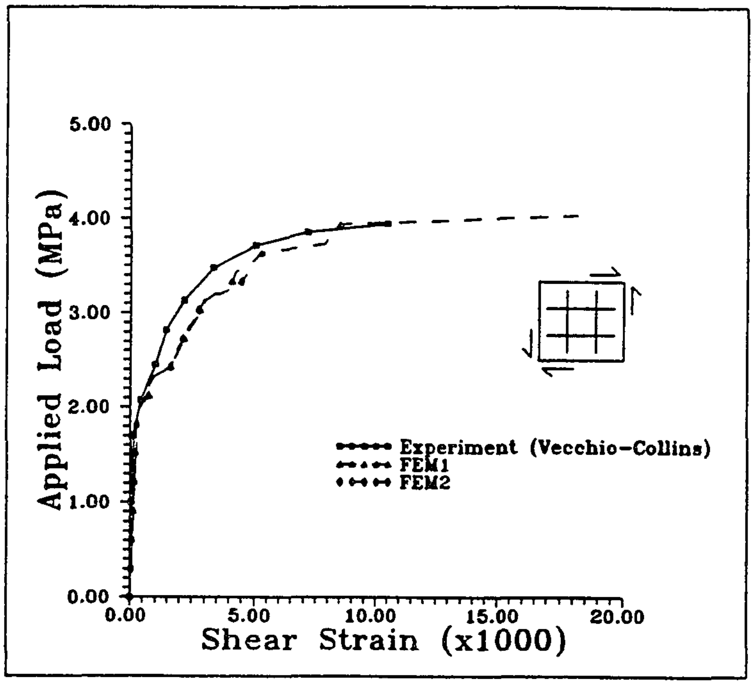


Figure 6.28: Load vs. Shear Strain Response, Specimen PV19

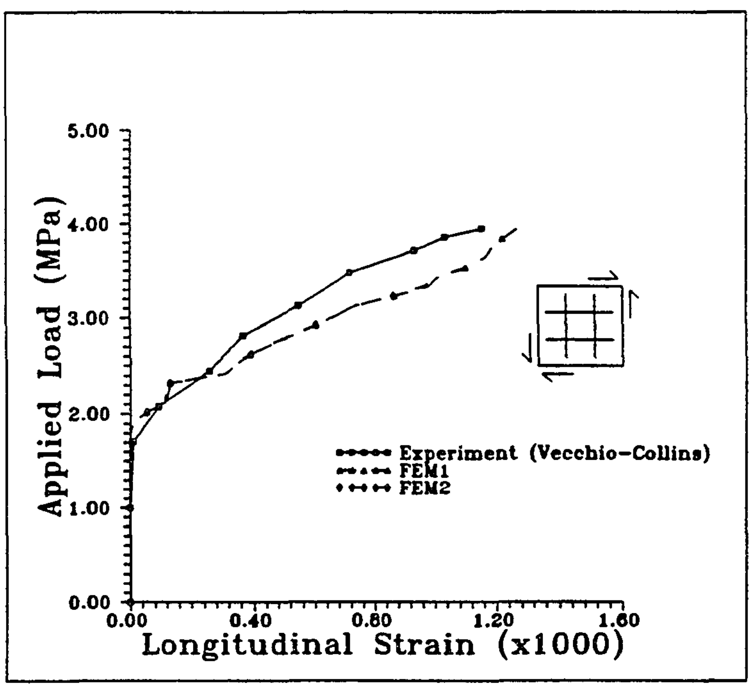


Figure 6.29: Load vs. Longitudinal Strain Response, Specimen PV19

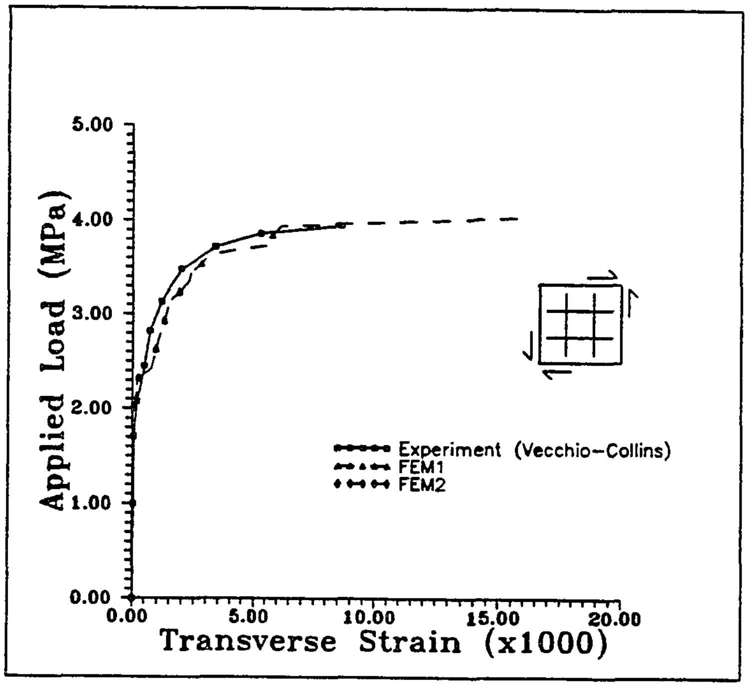


Figure 6.30: Load vs. Transverse Strain Response, Specimen PV19

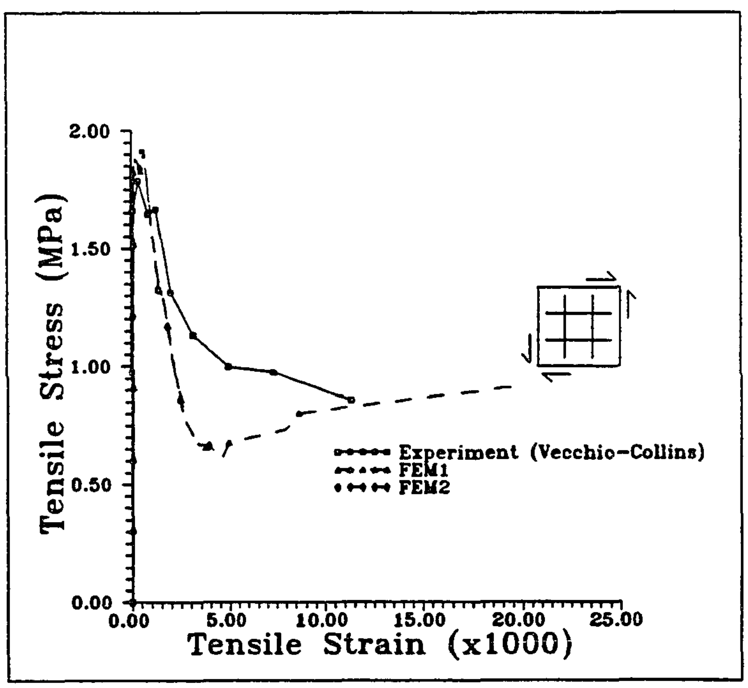


Figure 6.31: Tensile Stress-Strain Response, Specimen PV19

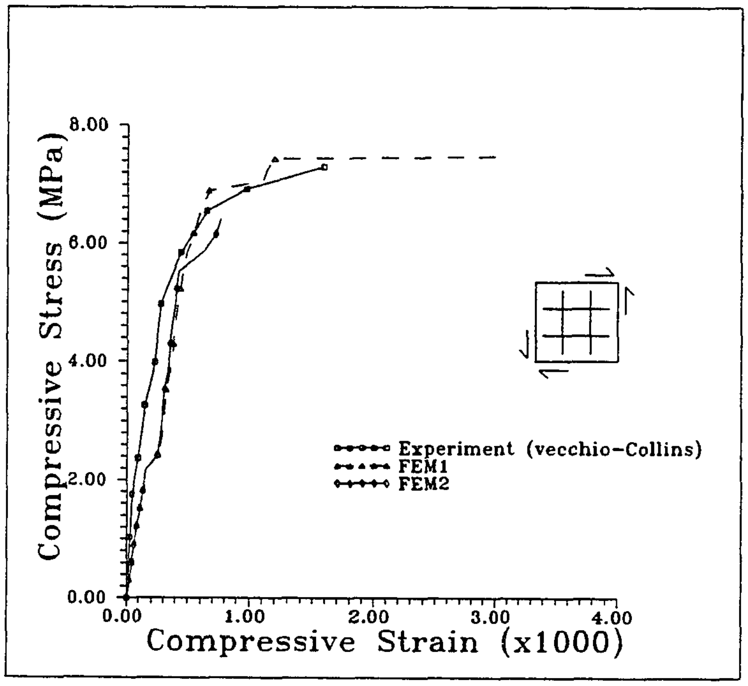


Figure 6.32: Compressive Stress-Strain Response, Specimen PV19

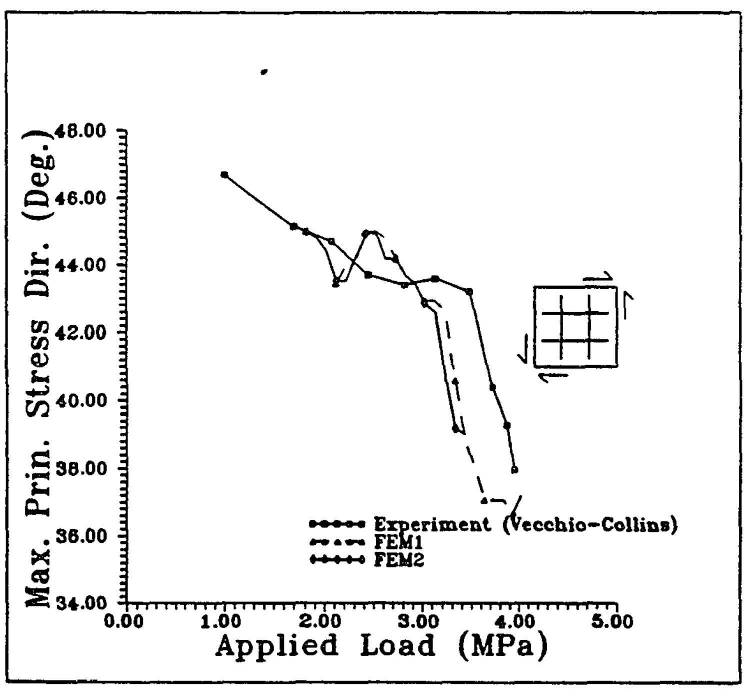


Figure 6.33: Principal Stress Direction vs. Axial Load, Specimen PV19

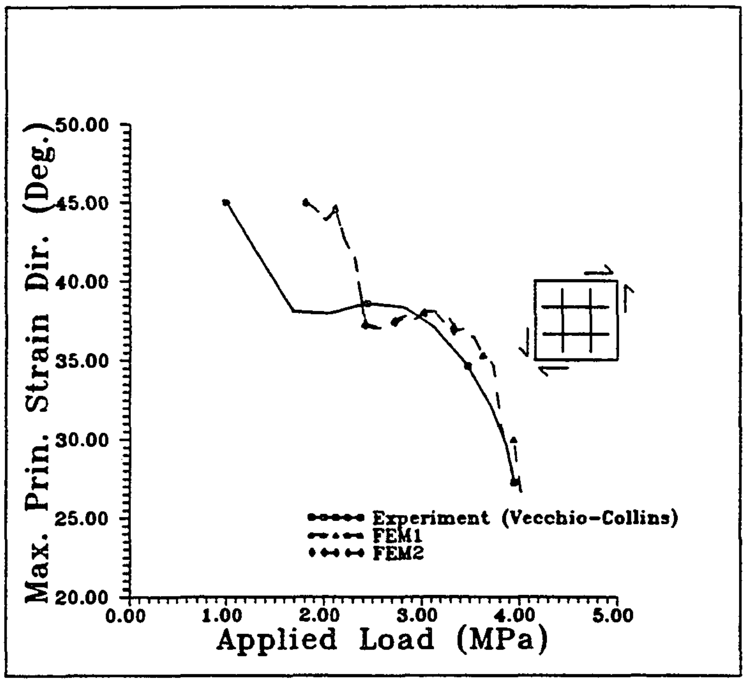


Figure 6.34: Principal Strain Direction vs. Axial Load, Specimen PV19

interesting to see that the analytical model FEM2, which has a strain-based post-cracking compression softening model, predicts the experimental response characteristics better than the stress based model FEM1. The strain components in the longitudinal and transverse directions are shown in figure 6.36 and figure 6.37 respectively. Both the analytical response simulations (FEM1 and FEM2) are almost identical and are more flexible than the experimental response. The tensile stress-strain response shown in figure 6.38 shows a much more rapid reduction of stresses in each of the analyses when compared with the experimental response. The behavior of the analytical models in the compressive stress-strain direction are quite different (figure 6.39). The analytical results FEM2 compare very well with those of the experiment, while the analysis FEM1 is very stiff.

A careful examination of the strength based failure criterion suggests that, with a rapid reduction of tensile stresses, larger compressive stresses are required to cause failure (crushing) as is seen in analysis FEM1. The strength reduction based on strains, is insensitive to the drop in tensile stresses observed in the cracked direction (figure 6.38). This suggests that some bond slip may be present, which also explains the occurrence of rather high tensile stresses observed in the experimental response even at high strain levels.

Panel PV29 : Nonproportional loading

PV29 is a an anisotropically reinforced panel element subjected to a nonproportional loading path. The panel was initially subjected to a pure shear load and then at a load of 3.9MPa a biaxial compressive load was introduced gradually along with the shear load. The tensile stress-strain paths shown in figure 6.40 for both the analyses FEM1 and FEM2 compare very well with the experimental response. It is seen that the tensile stresses are nearly exhausted and the nonproportional load applied at this late stage does not changing the behavior. However the strains in the transverse steel direction continue to grow very rapidly in the experiment, as seen from figure 6.41, even beyond yielding of steel. In the analysis the application of the compressive loads was found to arrest the strain increase and no yielding was registered. The strains in the longitudinal steel direction (which was the stronger direction) showed much better correlation with the experiment (figure 6.42). Both the compressive stress-strain response (figure 6.43) and the shear stress-strain response (figure 6.44) indicate a stiffening after the application of the compressive loads. The reduction of tensile stresses causes the compressive stresses and the ultimate load to be much larger in the analysis FEM1 (ultimate load of 7.1MPa) when compared to the experimental results (5.8Mpa). The analytical response FEM2 also appears to have a stiffer response and larger ultimate load(6.9MPa). The inherent path independency of the analytical model could be a reason for some differences. One possible explanation for the observed experimental response behavior is an improvement in the bond transfer capacity due to the application of the lateral compressive loads. The analytical studies are calibrated based on uniaxial test data without any lateral compressive loads present. An increase in bond transfer would possibly cause the transverse steel direction to yield as observed in the experiment, thus making the response more flexible.

Panel CS6: Non-proportional Loading

RC panel CS6 is a bi-axially reinforced panel, subjected to non-proportional loads.

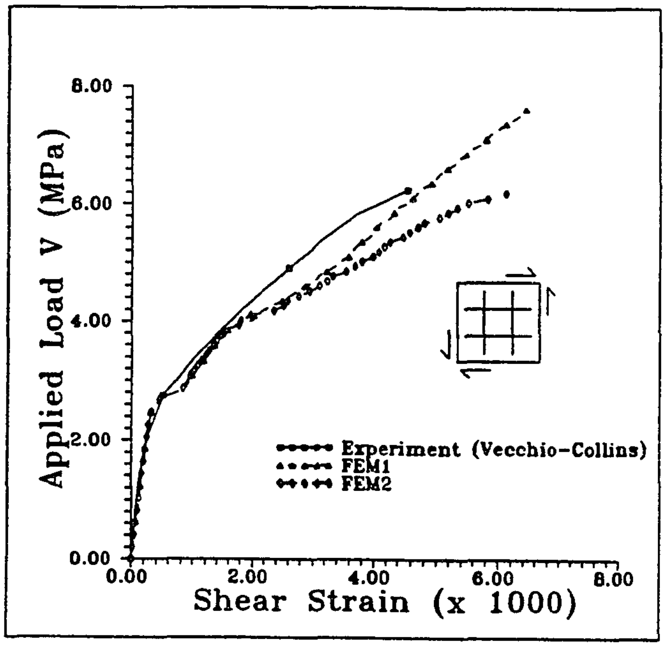


Figure 6.35: Load vs. Shear Strain Response, Specimen PV27

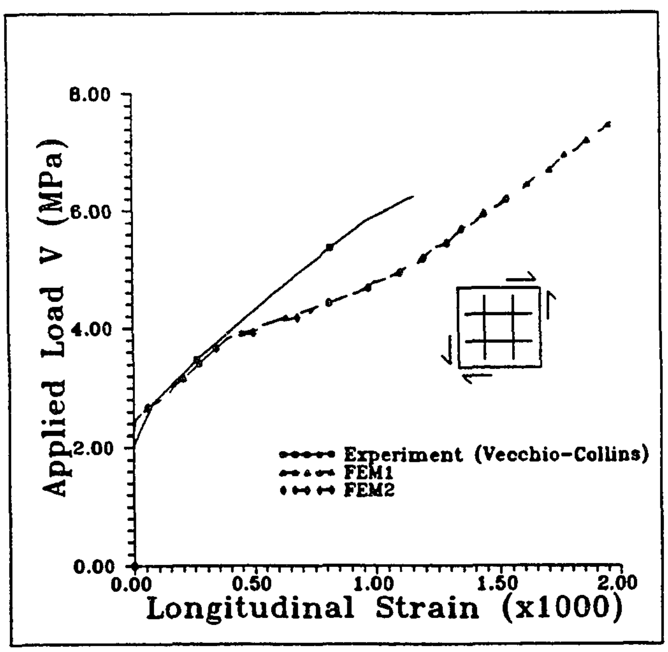


Figure 6.36: Load vs. Longitudinal Strain Response, Specimen PV27

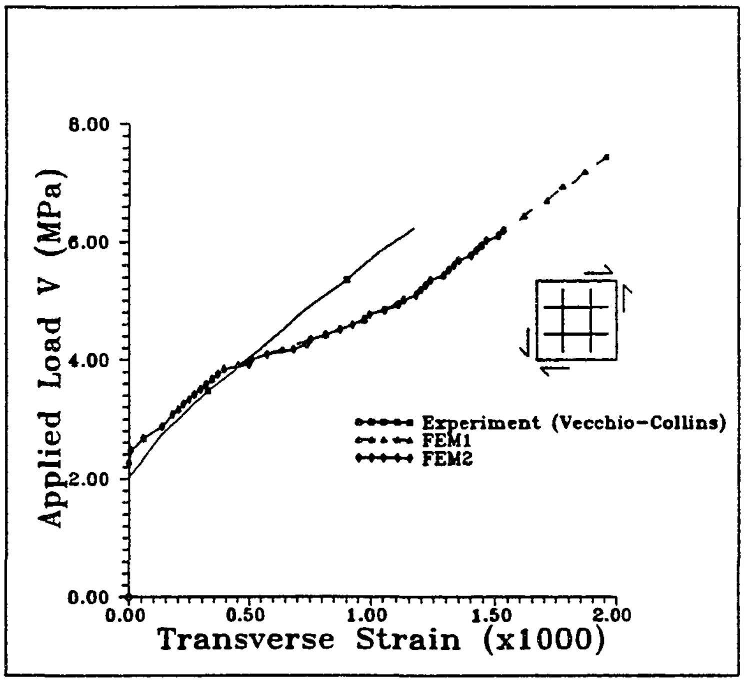


Figure 6.37: Load vs. Transverse Strain Response, Specimen PV27

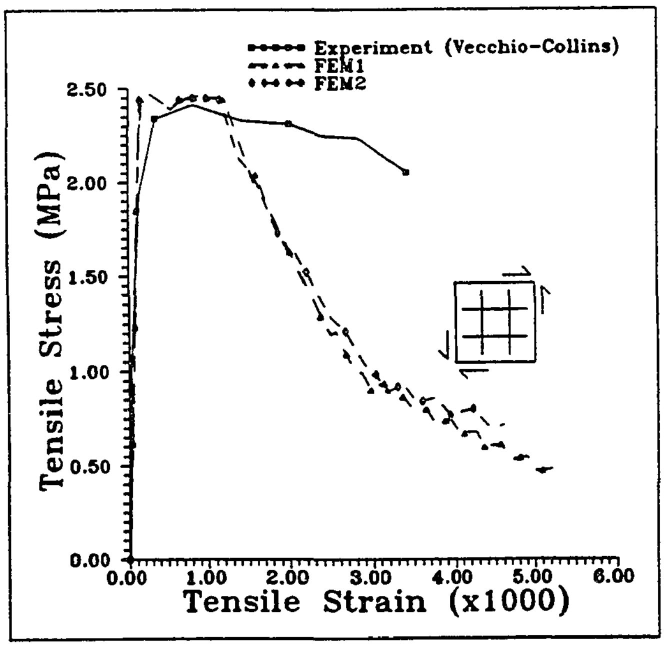


Figure 6.38: Tensile Stress-Strain Response, Specimen PV27

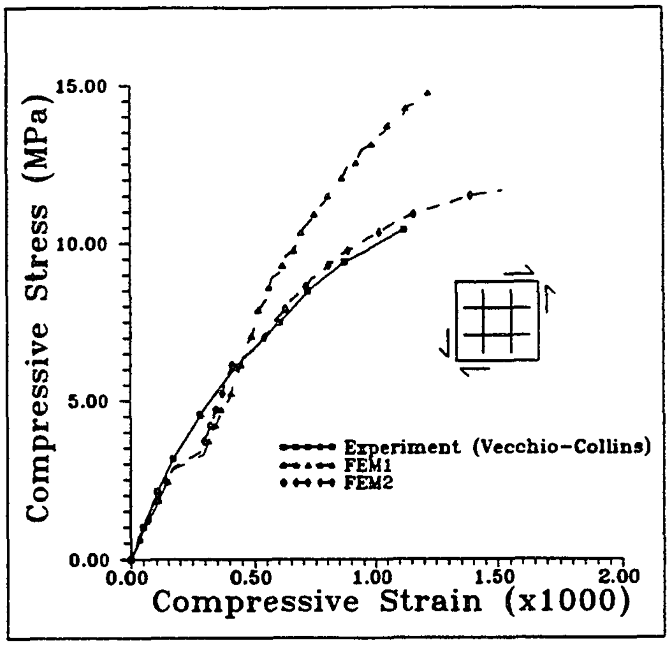


Figure 6.39: Compressive Stress-Strain Response, Specimen PV27

The specimen is initially subjected to cyclic tension along the transverse steel direction, and then subjected to compression in the longitudinal direction keeping the tension in the transverse direction constant. The details of the geometry, material properties, and loading of this panel, tested by Dyngland (1989), is given in tables 6.6 and 6.7. The tensile and compressive stress-strain response of this test specimen is shown in figure 6.45 and 6.46 respectively. While in the experiment the specimen was subjected to cycles of tensile loading, the analysis has been restricted to monotonically increasing loads up to the experimental value (stage 1). From figure 6.45, it is observed that the experimental tensile stress-strain response, excluding the unloading/ reloading behavior, is similar to the analytically obtained results. On the application of compression in the longitudinal direction the strains continue to increase in the transverse direction (stage 2) . This behavior clearly indicates that some 'Poisson's effect' is present even in cracked reinforced concrete and must be accounted for in analytical models. In the present study, this response has been obtained through the use of post-cracking coupling formulation described

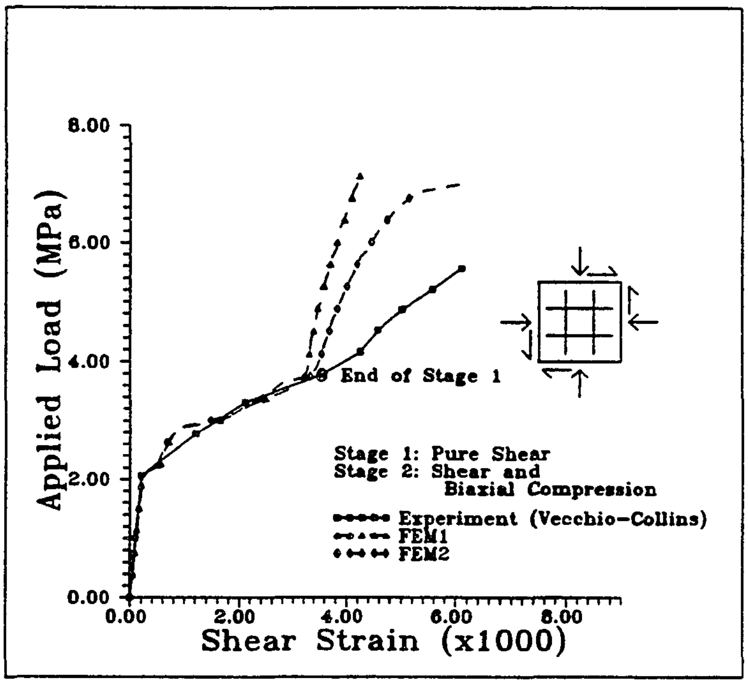


Figure 6.40: Load vs. Shear Strain Response, Specimen PV29

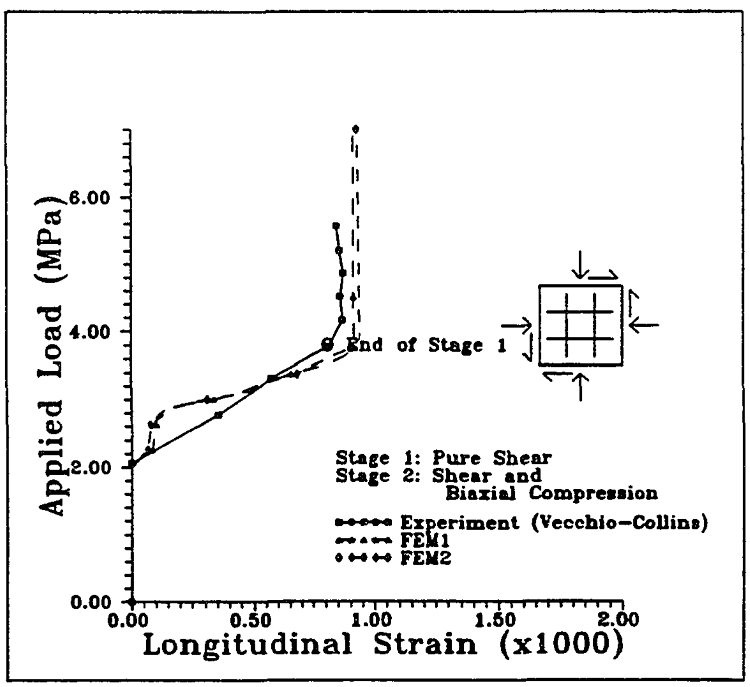


Figure 6.41: Load vs. Longitudinal Strain Response, Specimen PV29

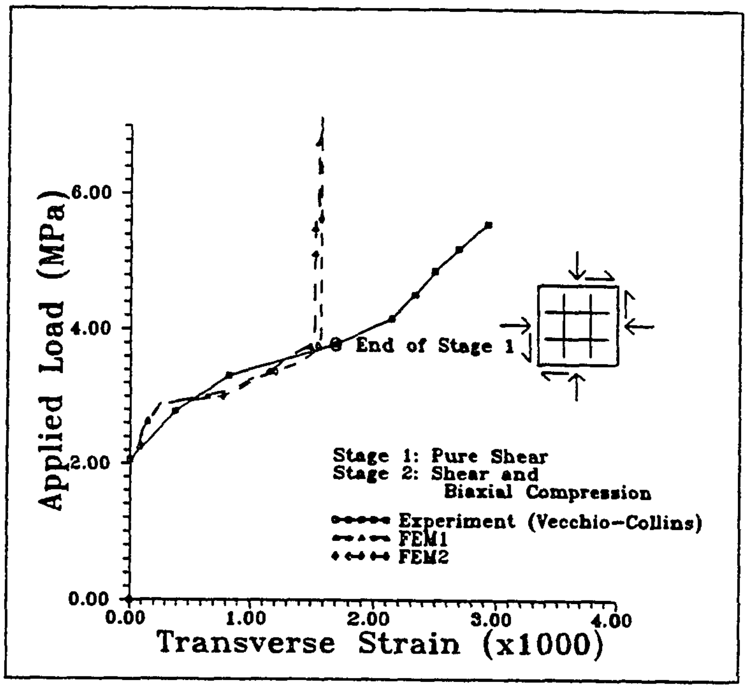


Figure 6.42: Load vs. Transverse Strain Response, Specimen PV29

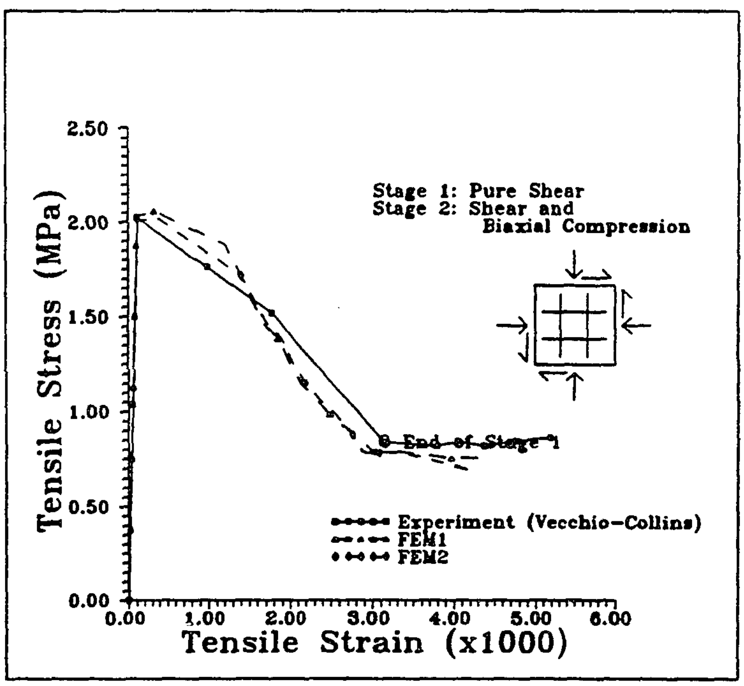


Figure 6.43: Tensile Stress-Strain Response, Specimen PV29

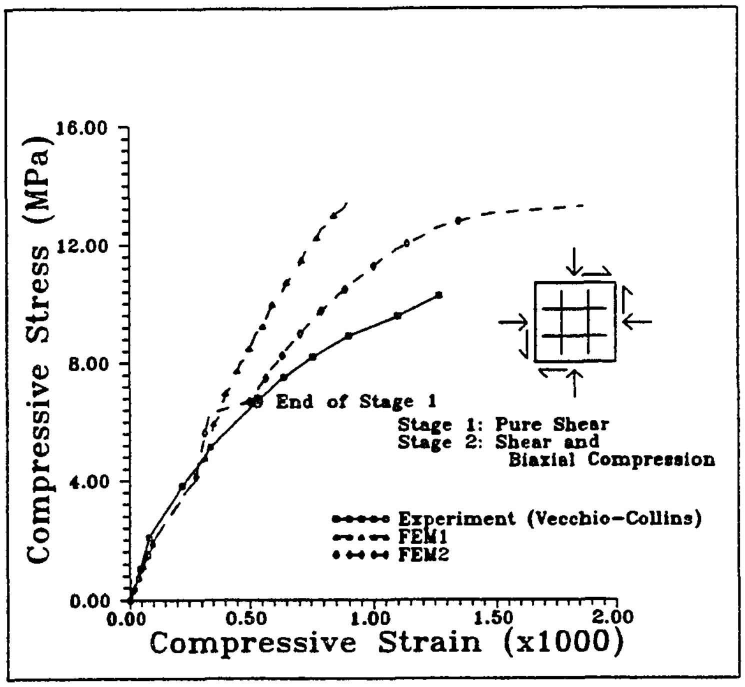


Figure 6.44: Compressive Stress-Strain Response, Specimen PV29

in chapter four. The analytical compressive stress-strain response prediction (FEM1 and FEM2) shown in figure 6.46 also compares well with the experimental results. The ultimate loads are shown in table 6.8. The nonproportional loading path in this test specimen appears to be well represented by the analytical model.

6.2.1.3 Summary

A number of panel specimen subjected to different membrane loadings have been analyzed to highlight the influence of the different post-cracking components of the analytical model, vis. cracking, tension-stiffening and its coupling effects, aggregate interlock and softening of the compressive strut. Based on the response predictions of these different specimen under uniform membrane loadings, the following observations can be made:

- The tension-stiffening model including its coupling effects is able to capture the response of most dominant characteristics reasonably well. The tensile

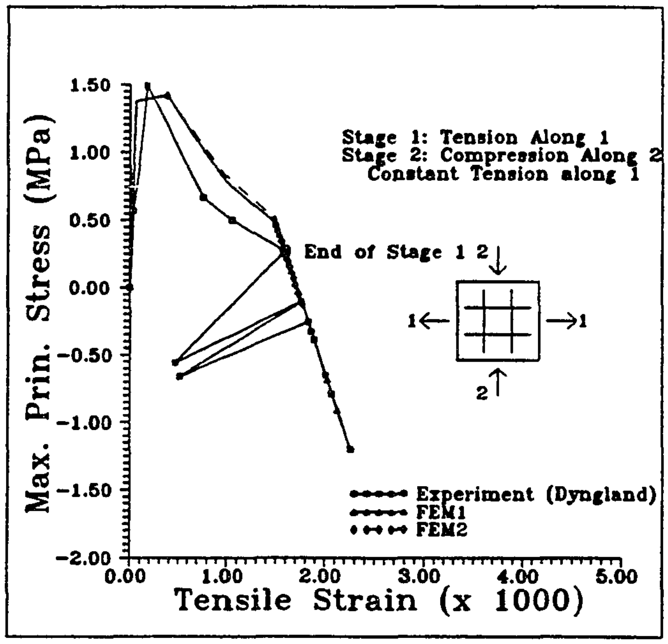


Figure 6.45: Tensile Stress-Strain Response, Specimen CS6

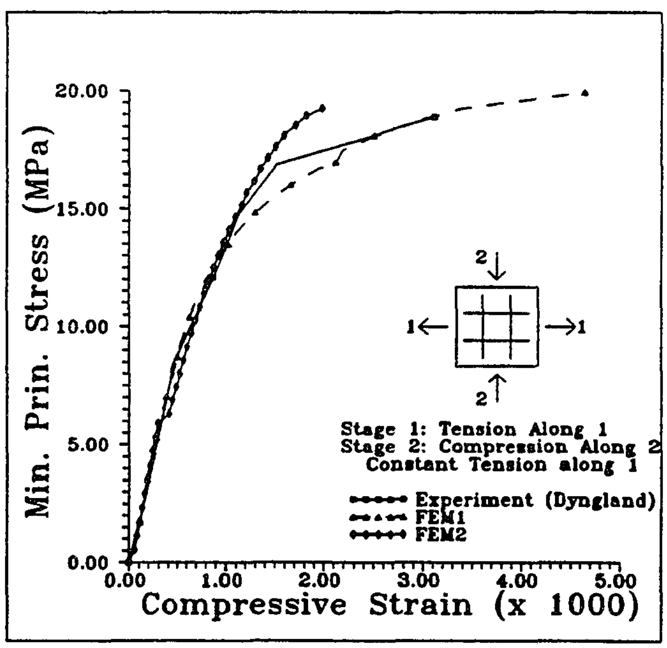


Figure 6.46: Compressive Stress-Strain Response, Specimen CS6

stress-strain responses of specimen in the Dyngland (1989) study (S1, S3, S5, S6, S7), Bhide-Collins (1987) study (PB18, PB19, PB20, PB21, PB22) and the Vecchio Collins (1982) (PV19, PV27, PV29) study are indicative of its capabilities. The parameters used to describe the shape of the tension-stiffening spring are reasonable and have been used for the remainder of the present study. The tensile stress-strain response prediction for specimen CS6, tested by Dyngland (1989), clearly illustrates capabilities of the model in capturing the coupling effects after cracking.

- The aggregate interlock model has been able to capture the sliding component of the deformational response adequately. The shear stress-shear strain response, the rotation of the principal stress direction and the transverse and longitudinal strain responses have all been well represented by the analytical model.

- The softening of the compressive strut has been examined using two different formulations. While the strain based softening formulation performs well in many of the analyzed specimens it has some inherent drawbacks. It suppresses the influence of bond-slip, e.g. specimen PV27, and predicts the compressive stress-strain response well, while the tensile stress-strain response prediction of the specimen is quite different from the experimentally obtained results. Strain based strength and stiffness reduction has also resulted in predicting premature crushing for some specimen, e.g. PV19. The stress-based softening of the cracked concrete compressive strut has resulted in stiff stress-strain responses in some problems. However, this formulation highlights the critical aspects of material response behavior reliably and has been employed in the analytical model for the remainder of this study.

6.2.2 Membrane Loading with Stress Gradients: Shearwalls

Reinforced concrete shearwalls are commonly used in high-rise buildings to provide resistance to wind and earthquake loads. Walls are categorized into two classes in the ACI code: short walls (Height to width ratio of one), that are primarily designed to mobilize sufficient shear strength to resist loads, and tall walls (Height to width ratios greater then 1.5) that are designed to sustain loads through their flexural capacities. The ACI code provisions are based on uniaxial strength properties of concrete. The amount of reinforcement required in the horizontal and vertical directions are based on a failure mode resulting in the tearing of the tension zone near the base of the wall.

Recent experimental studies of shearwalls by Maier and Thurlimann (1985), and Leifas et al. (1990a) indicate that failure of these walls was due to significant spalling of concrete near the compression toe of the wall. To investigate the influence of the various parameters on the strength of shearwall elements, Leifas et al. (1990a) varied a number of parameters between tests such as the uniaxial strength of concrete, percentages of vertical and horizontal reinforcement, and the amount of dead load present (Vertical loads were applied initially and kept constant) at the time of applying shear loads. From these tests, they concluded that the uniaxial strength properties of concrete was not sufficient to compute the flexural capacity of walls.

As expected the presence of vertical loads in the wall elements stiffened the horizontal load-lateral displacement response of the wall and also increased the ultimate load capacity of the walls. However, it did not seem to alter the mode of failure in these wall elements which was one of failure in the compressive toe region.

Analytical investigations of wall elements, by Wang et al. (1989) indicate that the failure of these elements was due to crushing in the compression toe. In order to obtain the ultimate load of the experiments, these analyses were conducted using modified material properties. The uniaxial strength of concrete was increased and only the material in the portions of the wall near the compression region was permitted to become nonlinear. The rest of the wall was treated as an elastic-cracking solid. This investigation concluded that some parameter other than cracking, post-cracking nonlinearities, and the yielding of the reinforcement was responsible for the observed increase in strength. Analytical studies by Leifas et al. (1990b), indicates that in order to predict the increased strength observed in these walls, additional reinforcement representing confinement of the concrete, was included in the edge elements. The ultimate load was underestimated without this artificial increase in the reinforcement. They concluded that the height-to-width ratio and the amount of vertical reinforcement was not properly accounted for in the current code recommended provisions for evaluating the shear strength of structural walls.

In the present study, four walls were selected, two from each experimental investigation. The material properties, dimensions of these specimen, and finite element mesh details are presented in tables 6.9 and 6.10 and figures 6.47 and 6.48. Instead of arbitrarily altering the strength or the geometric properties to obtain the experimentally observed ultimate load, emphasis was placed on isolating the dominant phenomenon leading to failure. The load-deflection response obtained in the analyses for each of the specimen are shown in figures 6.49 to 6.52. The experimental and analytical ultimate loads for these specimen are compared in table 6.11. The results indicate that the ultimate strength of the walls obtained in the analysis underestimate the experimental values by as much as 30%. The unconverged point indicated in the analysis (figure 6.49 to 6.52) is due to excessive crushing in the compression toe region which resulted in the inability of the analysis to continue the solution procedure. The deflected shape of the specimen is shown in figure 6.53. The mode of failure obtained in the analysis is by crushing in the compression toe. The compressive stress distribution within the specimen near the ultimate load is shown in figure 6.54. The compressive stresses near the toe were at about the biaxial compressive strength levels (1.16 f_c'). The cracking patterns near the ultimate load are shown in figure 6.55. The width of the flexural compressive zone reduces very rapidly near the ultimate. The horizontal and vertical steel stress distributions are shown in figures 6.56 and 6.57, respectively. The vertical steel had yielded in both tension and compression, while the horizontal steel stresses was quite small.

One of these wall specimen, SW21, was selected for a parametric study. The parameters investigated in this study were the different compression softening models in cracked concrete, the threshold angle for multiple cracking (changed from 30^o to 90^o), and the boundary condition at the base of the beam (changed from a fixed

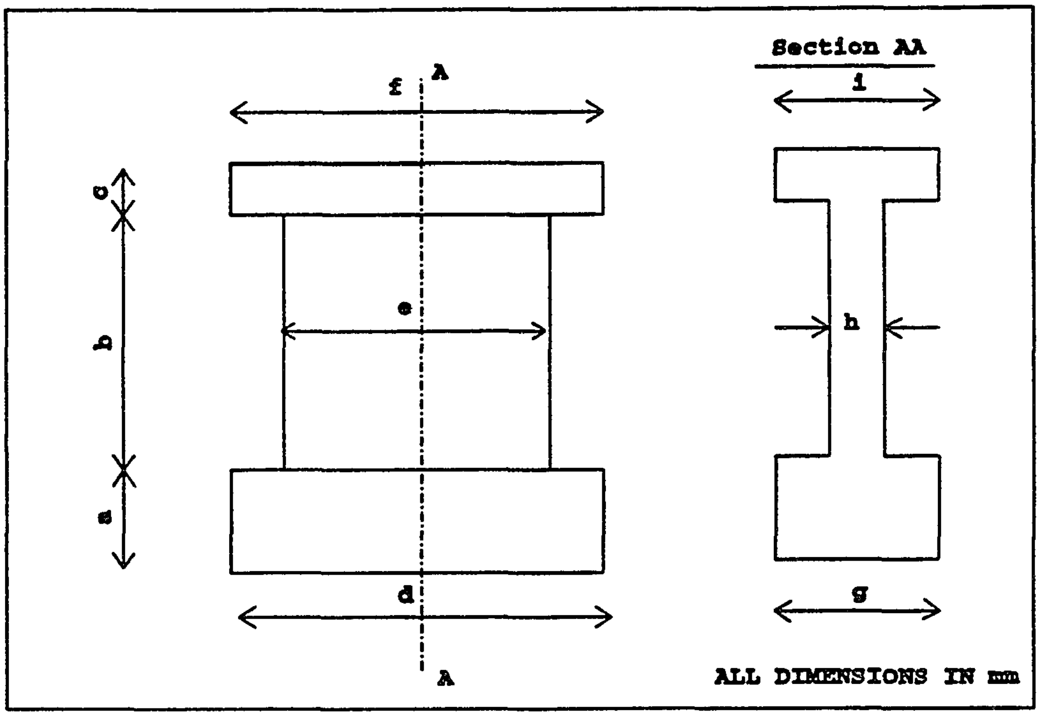


Figure 6.47: Dimensions of Shearwall Test Specimen

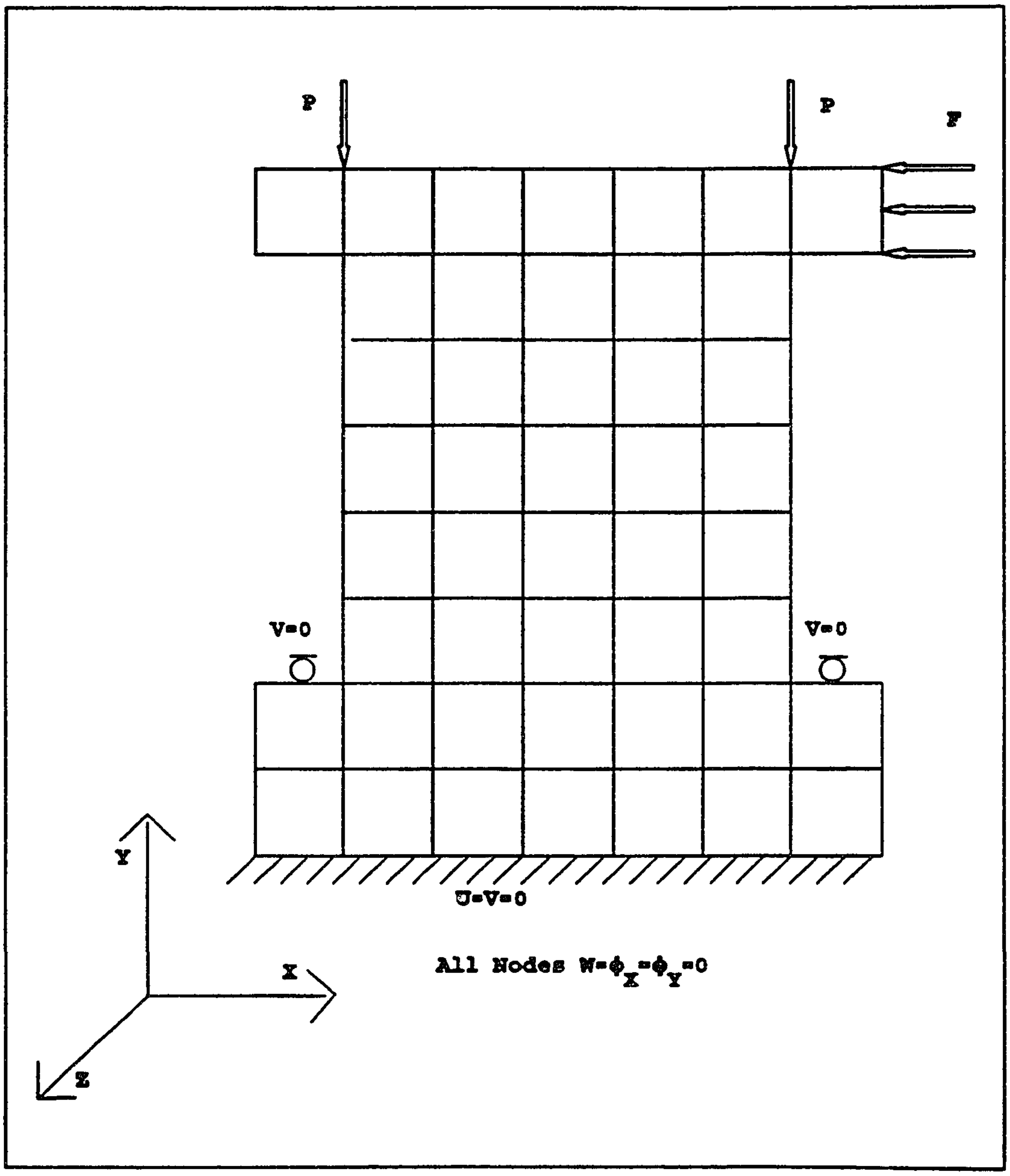


Figure 6.48: Finite Element Mesh For Shearwall Specimen

base to a simple bolting of the beam to the base at four points, thus permitting movement at all other points). The results of these analyses are also shown in figure 6.50. It is seen that none of these parameters appear to have influenced the load deformation-response or the ultimate mode of failure.

Table 6.9: Dimensions of Shearwall Test Specimens

Specimen	Dimensions (mm)								
	a	b	c	d	e	f	g	h	i
SW11	300	750	150	1150	750	1150	200	70	200
Sw21	300	1300	150	1050	650	1050	200	65	200
Maier#4	250	1120	250	0	1180	0	0	100	0
Maier#9	250	1180	250	0	1180	0	0	100	0

Table 6.10: Material Properties of Shearwall Specimens

Speci.	Concrete			Reinforcement			
	f'_c (MPa)	ϵ'_c $\frac{mm}{mm}$	f'_t (MPa)	$\rho_{horiz.}$ (%)	$f_y^{horiz.}$ (MPa)	$\rho_{vert.}$ (%)	$f_y^{vert.}$ (MPa)
SW11	44.5	0.002	4.	1.1	520	2.4	470
Sw21	36.38	0.002	4.	0.8	520	2.5	470
Maier#4	32.9	0.0023	3.4	1.03	565	1.05	565
Maier#9	29.2	0.0023	3.2	0.0	0.0	0.99	565.

The experimentally obtained strength of these wall specimen appear to have been significantly enhanced possibly due to the development of triaxial compressive stresses in the compressive toe region. One possible explanation for the development of these stresses is the influence of the monolithic beam at the base of the wall resisting the development of lateral strains in the wall near the compression toe region. The triaxial stresses developed due to this could cause the increased strength and failure due to crushing and spalling of the concrete in the direction of the least compressive stresses. Another possibility is the presence of confining reinforcement in the beam and the wall edges which could contribute to the triaxial stresses. Even

Table 6.11: Results of Tests on Shearwalls

Speci.	Experiment		FE Analysis		ACI-318-89		
	Load (KN)	Failure Mode	Load (KN)	Failure Mode	Flexure Strength f_{su}(KN)	Shear Strength	
						V_c+ V_s(KN)	V_s (KN)
SW11	260	crushing	182	crushing	249.2	387.9	300.3
SW21	127	crushing	83	crushing	107.5	218.4	175.8
Maier#4[1]	370	crushing	305	crushing	302.9	930	550
Maier#9[1]	325	crushing	268	crushing	302.9	370	0
1:A vertical load of 260KN is applied initially and kept constant							

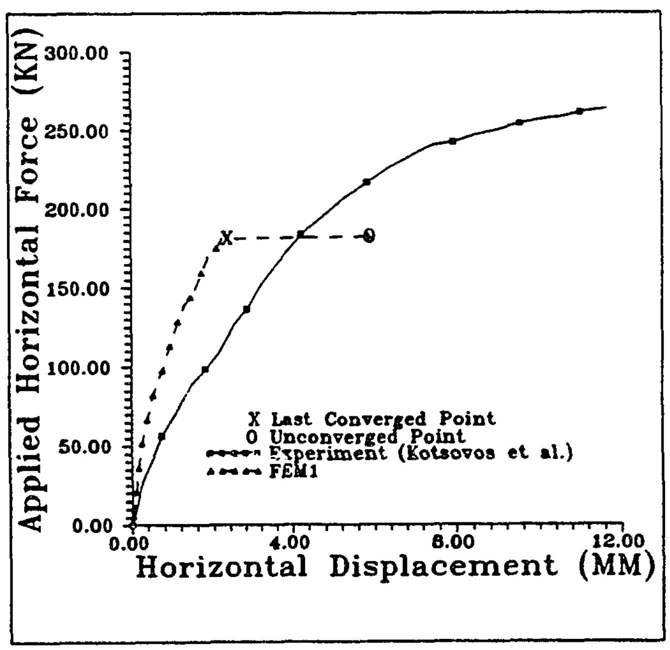


Figure 6.49: Load Displacement Response of Shearwall Specimen SW11

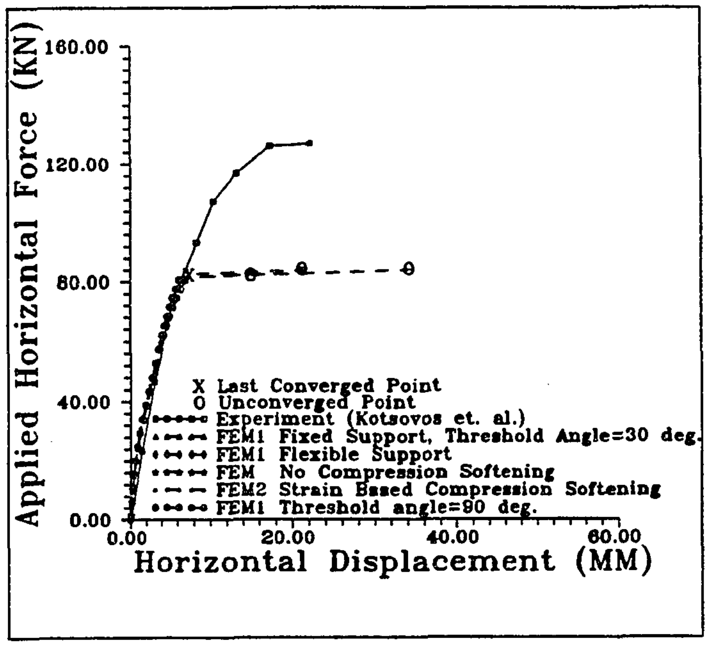


Figure 6.50: Load Displacement Response of Shearwall Specimen SW21

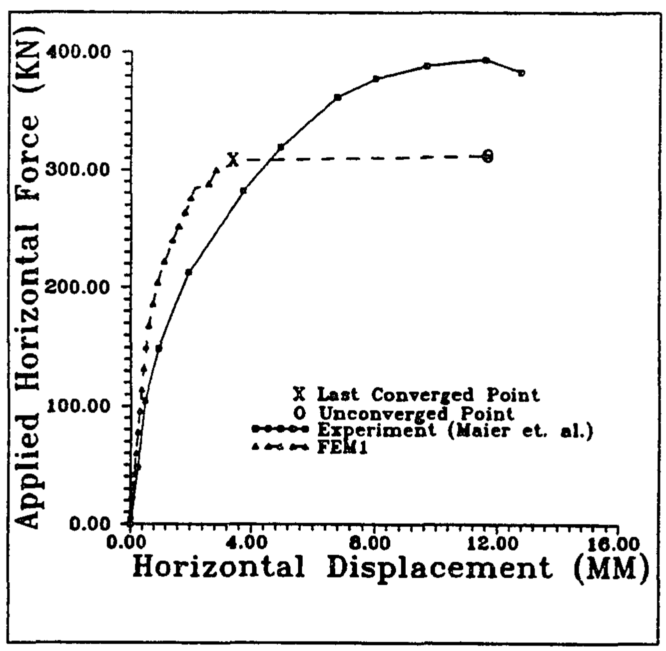


Figure 6.51: Load Displacement Response of Shearwall Specimen Maier#4

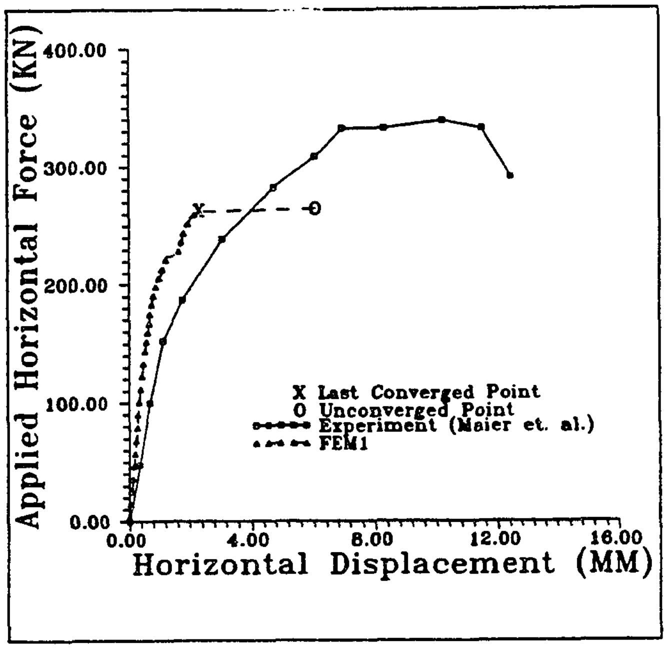


Figure 6.52: Load Displacement Response of Shearwall Specimen Maier#9

a small compressive stress of $5\% f'_c$ in the least compressive stress direction could increase the strength (triaxial compressive strength) of the specimen by as much as 20% due to the triaxial state of stress. It has been speculated (Kotsovos, 1988) that the dilatation present near compressive failure at some points would be resisted by the neighbouring points that are cracked, enhancing the strength of the wall element. Another factor that may be partially responsible for the underestimation of the ultimate load of the walls is the strain hardening in steel. However, complete data on the strain hardening properties of the reinforcement were not available and therefore not included in the analysis.

The ultimate strength predictions of these walls, in flexure and shear, using the ACI code are also shown in table 6.11. These results indicate that the estimated flexural capacity of these wall specimen, based on the ACI code, is exceeded in these tests. The contribution of the vertical reinforcement in sustaining the flexural loads has not been well accounted for in the ACI code. Further, truss analogy does not appear to be well suited for the analysis and design of shearwalls. The shear

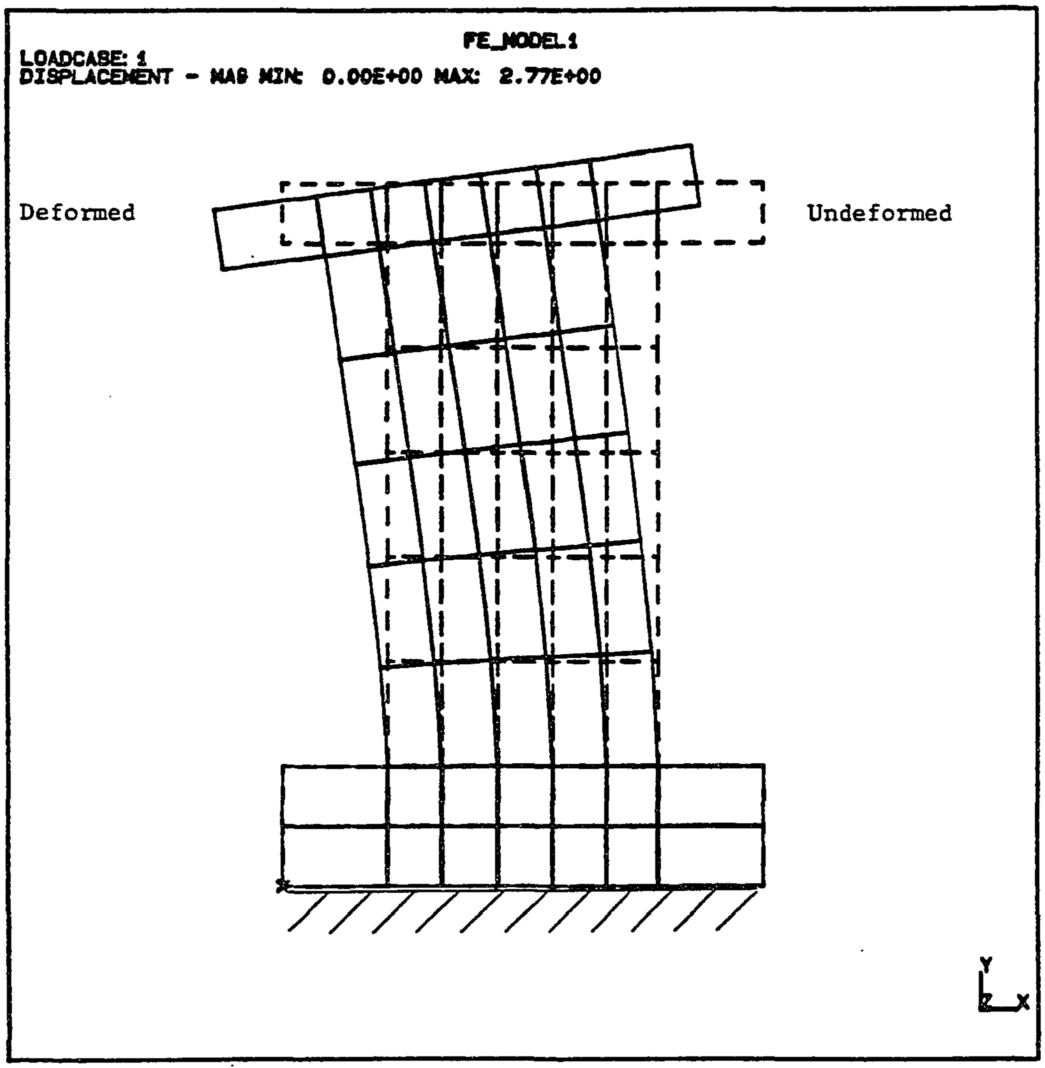


Figure 6.53: Deflection Profile, Specimen SW21

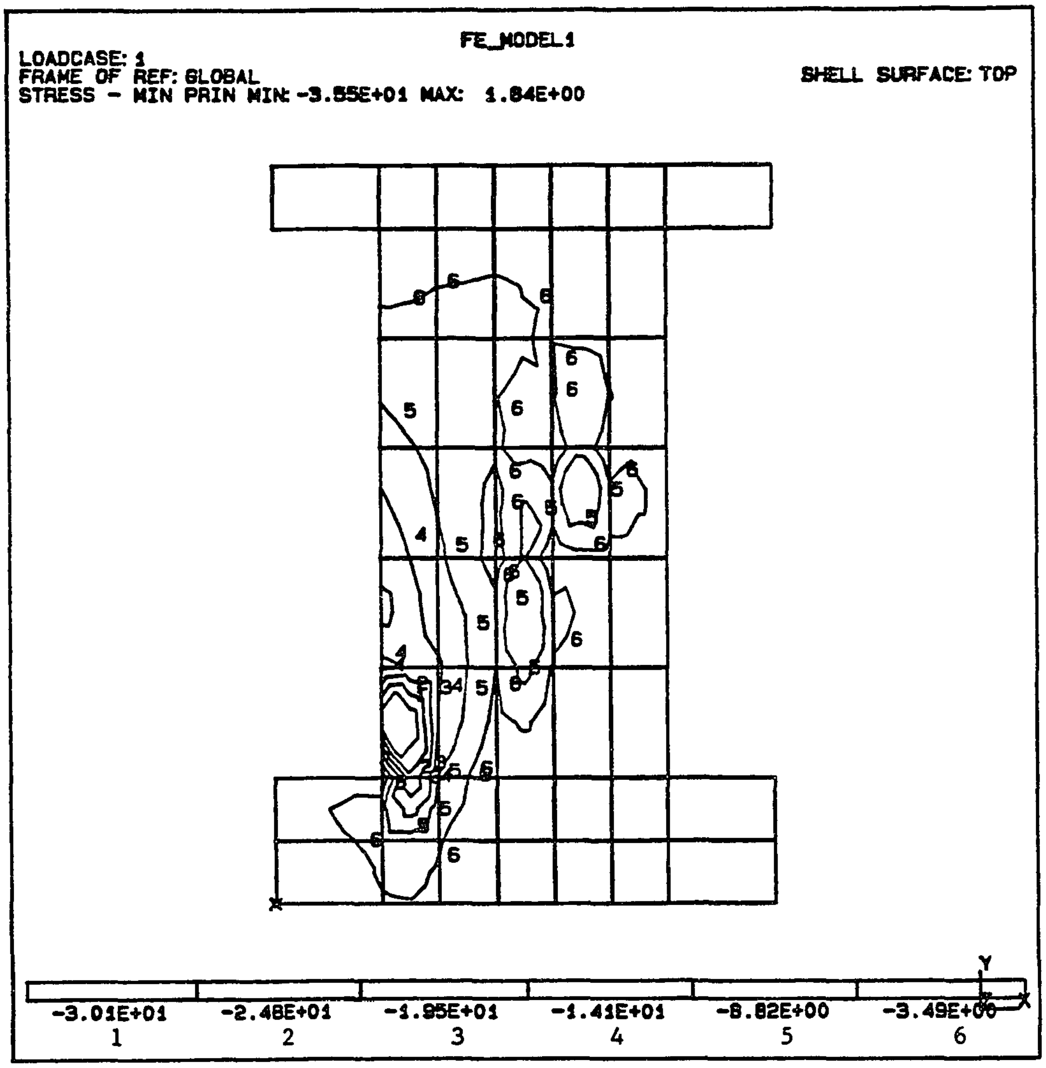


Figure 6.54: Compressive Stress Distribution in Shearwall Specimen SW21 Near Ultimate Load

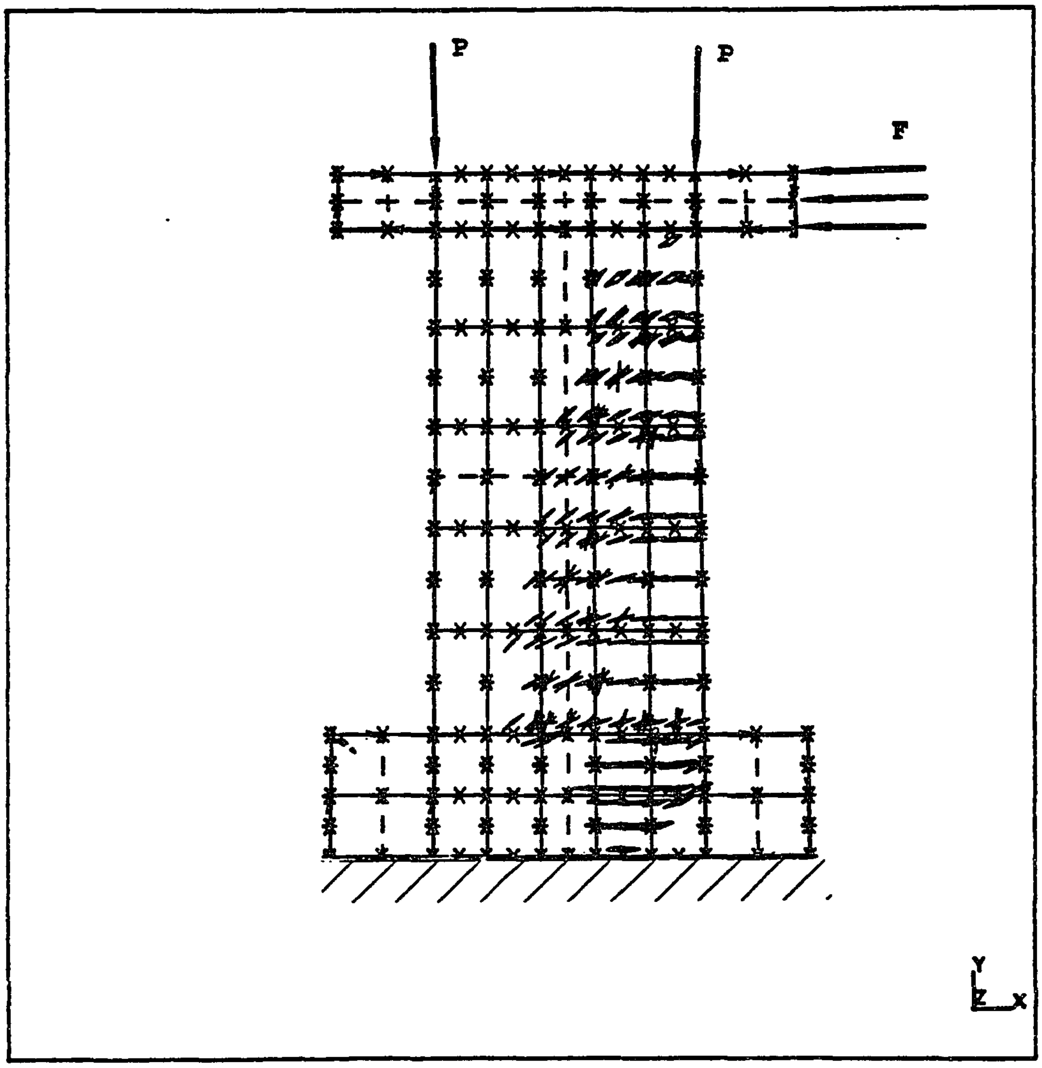


Figure 6.55: Crack Pattern in Shearwall Specimen SW21 Near Ultimate Load

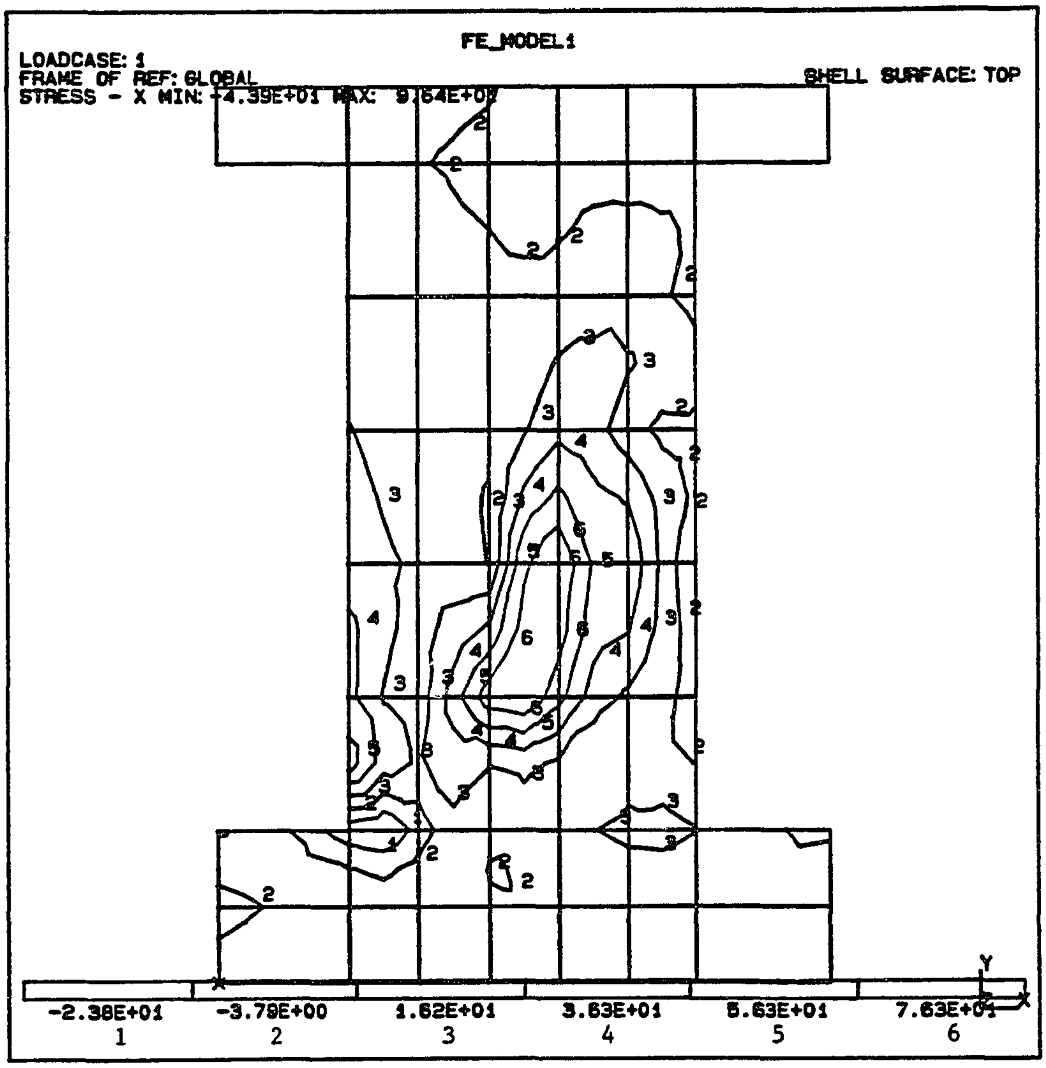


Figure 6.56: Stress Distribution in Horizontal Steel Near Ultimate Load, in Shear-wall Specimen SW21

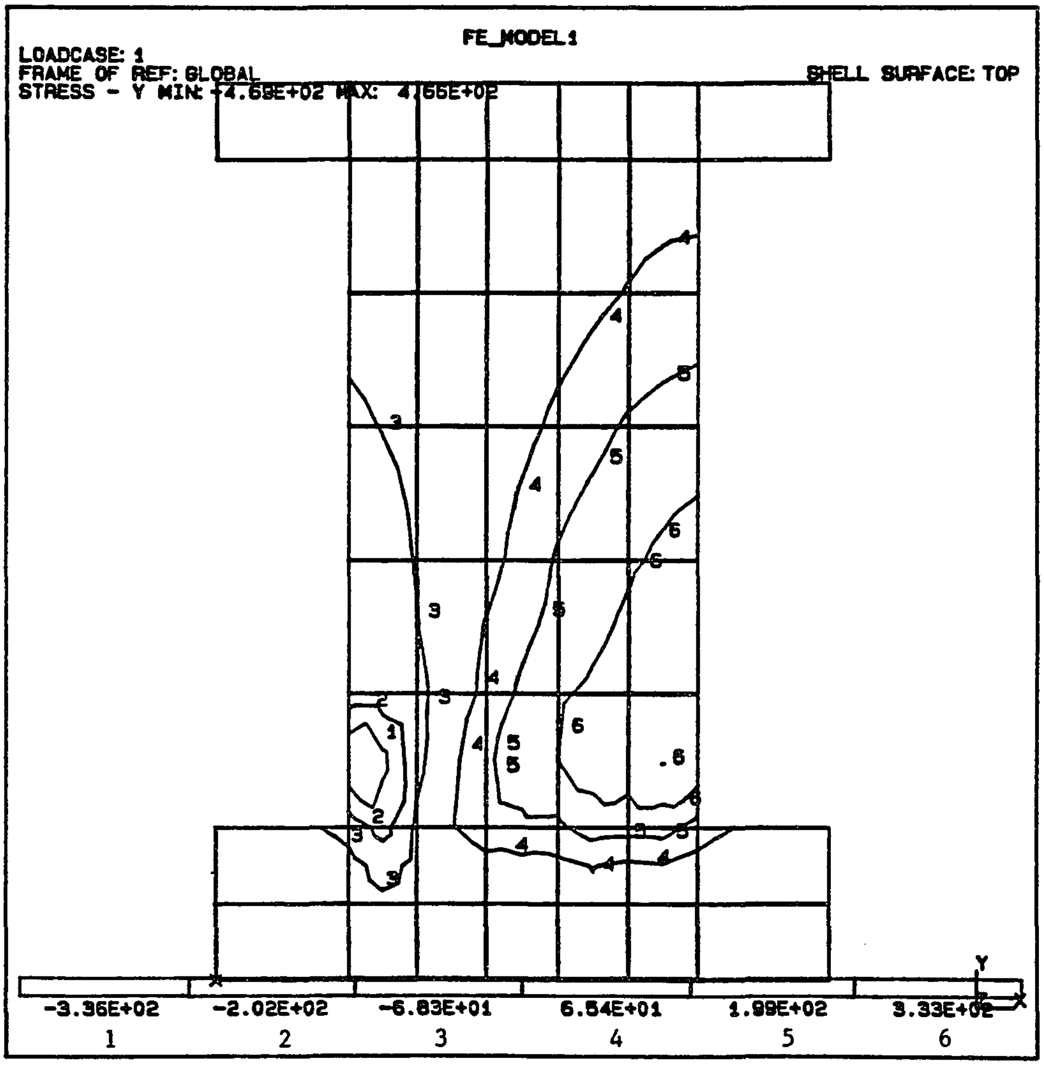


Figure 6.57: Stress Distribution in Vertical Steel Near Ultimate Load, in Shearwall Specimen SW21

capacity of these walls, predicted by the ACI code, considerably overestimate the strength of these wall specimen.

6.3 RC Elements Subjected to Out-of-Plane Loadings

Delft Beam

This moderately deep beam, tested experimentally by Walraven (1978), belonged to a group of beams tested to study the influence of beam depth on the shear strength of concrete. The beam was simply supported at the ends and subjected to concentrated loads at points 500mm on either side of the center. In the experiment it was observed that vertical cracks initially formed in the pure moment zone of the beam. As the loads were increased, new inclined cracks formed between the support and the loading point. It was reported that the final failure of the beam was due to the penetration of these inclined cracks deep into the compression zone resulting in a sudden failure. The material properties have been given in table 6.12. The beam geometry details are shown in figure 6.58 and the finite element mesh, along with the layering details are shown in figure 6.59.

In the present study this beam was analyzed using both the implicit layering procedure and the explicit layering procedure discussed in section 5.5. For the implicit layering analysis eleven layers, ten concrete layers and one steel layer, were used to discretize the cross-section. The concrete layers adjacent to the steel layers were tension-stiffening layers (each 15mm thick) while the other layers were strain-softening layers. The fracture energy G_f used in each strain-softening layer was 60N/m (deBorst and Nauta, 1985). The number of layers selected to discretize the cross-section is based on a previous study by Abdel-Rehman (1981), which concludes that between six to ten layers was sufficient to capture the response behavior reasonably. For the first analysis a relatively coarse mesh of seven elements along the length was selected to model half of the beam, taking advantage of symmetry (figure 6.59), using the implicit layering procedure. A 3x3x1 Gaussian numerical integration procedure has been employed to compute the stiffness of both the steel and concrete layers. The load vs. deformational response (center point deflection) is shown in figure 6.60 along with the experimental results, and results from an analytical study by deBorst and Nauta (1985). It was found that at a load of nearly 70 KN the beam 'failed' due to lack of convergence. The analytical response obtained by deBorst and Nauta (1985) indicated a much higher load carrying capacity (nearly 80 KN). They observed that in obtaining these results it was necessary to restrict the threshold angle required for the formation of multiple cracks at a point to 60°. The use of lower threshold angles (30°) resulted in a lack of convergence of the nonlinear procedure beyond a load of 70 KN. In the analysis with a refined mesh, deborst and Nauta (1985) observed a small reduction in the stiffness at a load of 70 KN but the ultimate failure was due to yielding of the reinforcement leading to a collapse of the tied arch mechanism. In the present study, a refined mesh (12 elements) was employed and the analysis was repeated. The results of this are also shown in figure 6.60. In this analysis, at approximately 70 KN, a kink in the load-deflection response was observed. On increased loading, this instability point was passed and the beam was able to sustain additional loads, failing eventually at 77 KN due to yielding of the steel. The deflected shape of the specimen is shown in figure 6.63. The cracking pattern for this analysis is shown in figure 6.66. The initial vertical cracks were in the pure moment zone, but when the loading was continued, new inclined cracks developed between the support and the loading point. These cracks penetrated deep into the compression zone near the ultimate load. The compressive stress distribution at the top surface of the concrete is shown in figure 6.64. The stress distribution in the reinforcement near ultimate load is shown in figure 6.65. The concrete in-plane normal and transverse shear stress distribution through the cross-section of the beam are shown in figures 6.61 and 6.62, respectively, at a point 1340mm from the support of the beam. As expected prior to cracking, the concrete normal stress distribution is linear while the transverse shear stress distribution is parabolic. At a load of 77KN (ultimate load), very small tensile normal stresses are present near the steel layer, and vanish completely in the lower two thirds of the beam cross-section due to cracking. Compressive stresses are mobilized in the top one third of the beam. It should be pointed out that the shear stresses plotted do not go up to the surface of the beam where they are expected to vanish.

Table 6.12: Material Properties of RC Elements Subjected to Out-of-Plane Loadings

Specimen	Concrete			Reinforcement $Mesh^1$			
	f'_c (MPa)	ϵ'_c $\frac{mm}{mm}$	f'_t (MPa)	ρ_{bottom} (%)	f_y^{bottom} (MPa)	ρ_{top} (%)	f_y^{top} (MPa)
ML9	44.4	0.0025	4.0	2	412	2	412
Delft $Beam^2$	35	0.002	2.5	0.77	440	0.0	0.0
Dudeck Slab	43.01	0.0027	2.0	0.69	670	0.297	670
Regan Slab1#2	24.8	0.002	2.5	1.2	500	0.0	0.0

1: Two way orthogonal reinforcement
2: Bottom steel along Beam length only

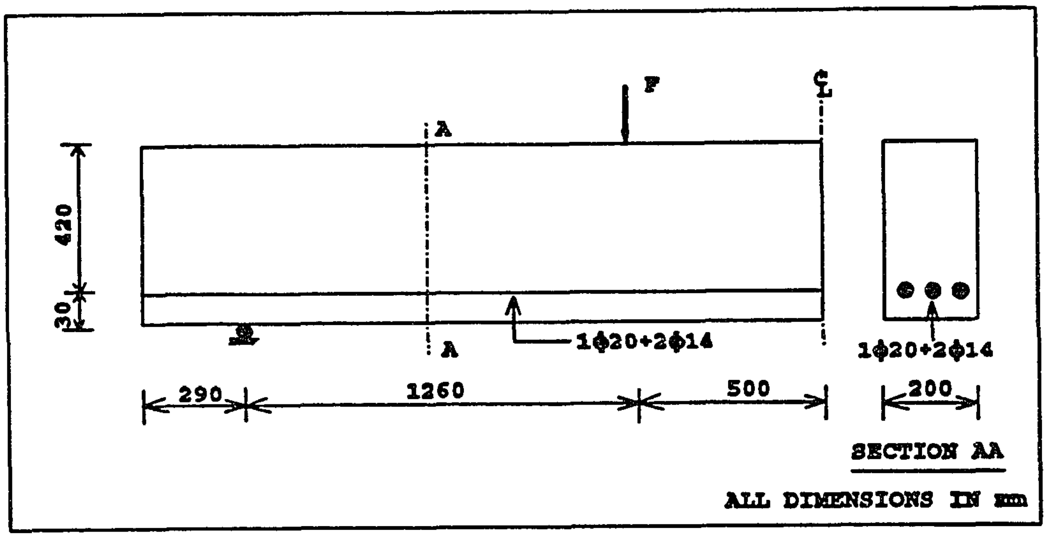


Figure 6.58: Dimensions of Specimen Delft Beam

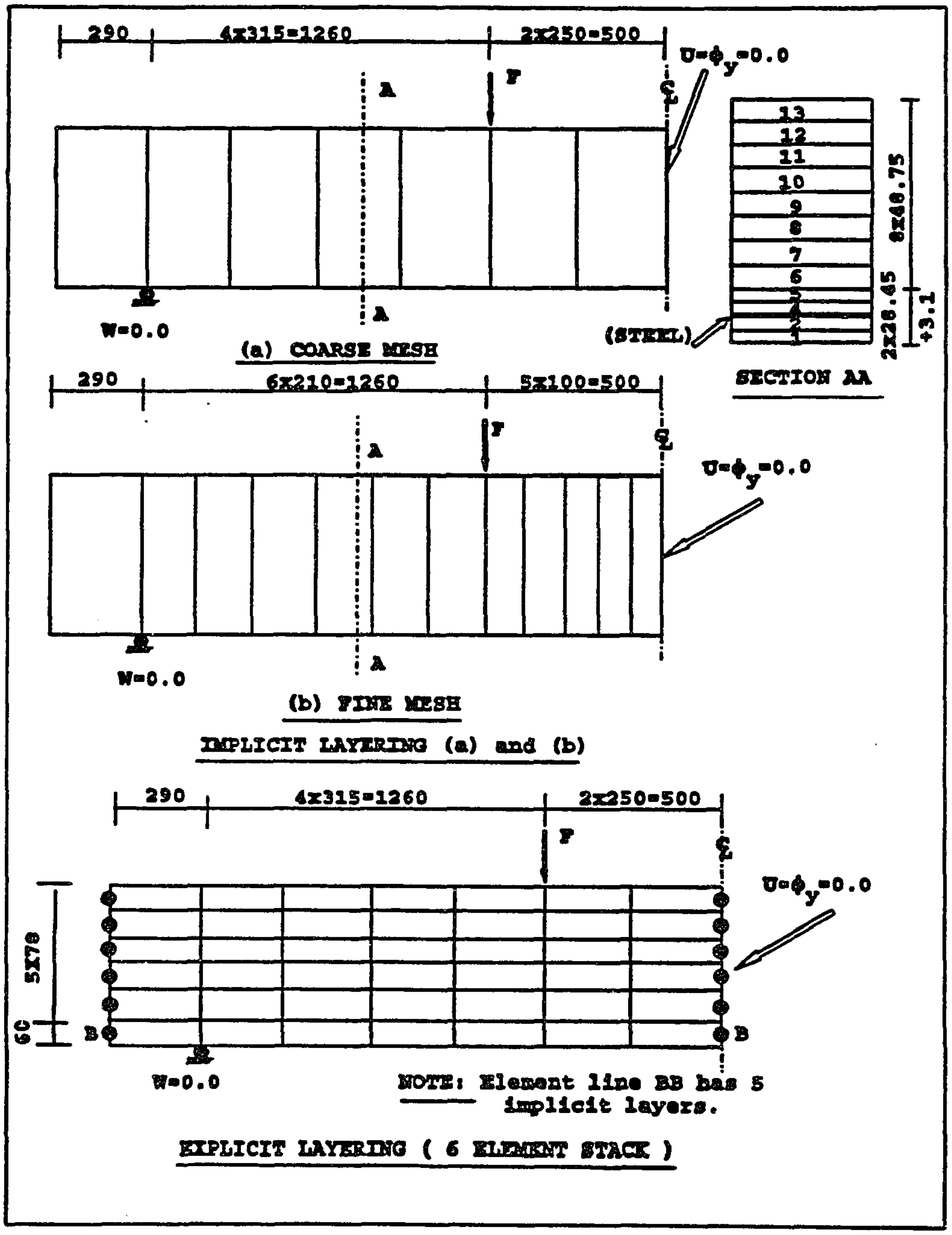


Figure 6.59: F.E. Mesh and Layering details for Specimen Delft Beam

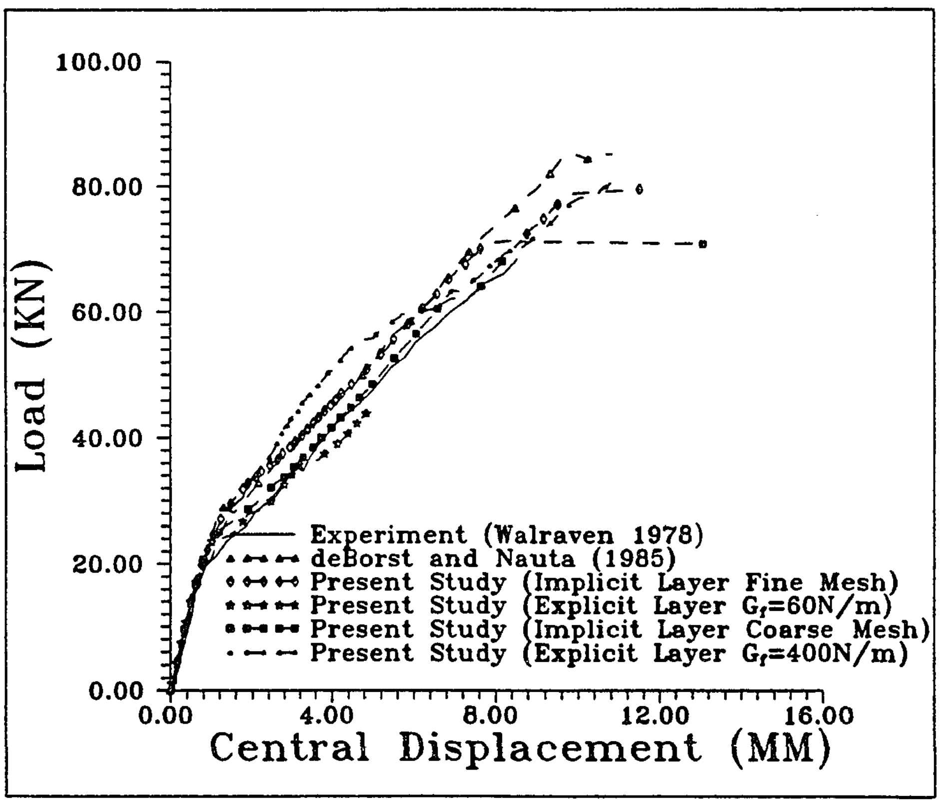


Figure 6.60: Load Vs. Deflection Response, Specimen Delft Beam

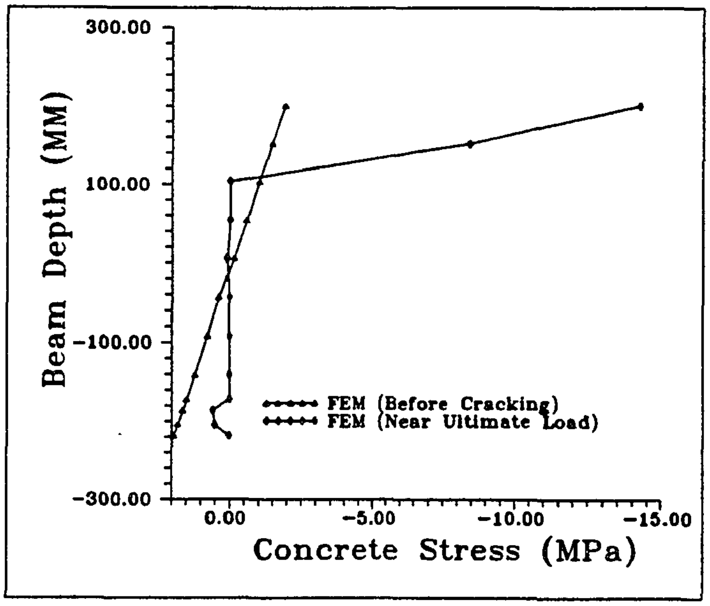


Figure 6.61: Normal Stress Distribution at Point, x=1340mm, from the Support, Implicit Layering Procedure

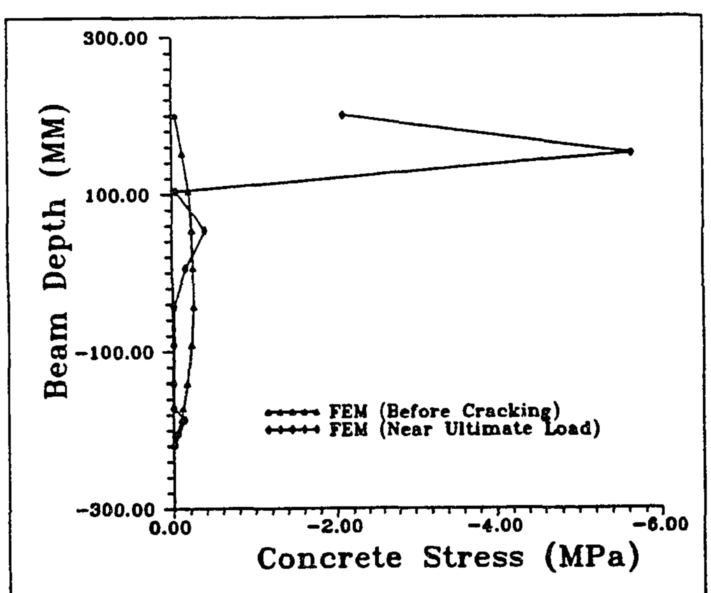


Figure 6.62: Transverse Shear Stress Distribution at a Point, x=1340mm, from the Support, Implicit Layering Procedure

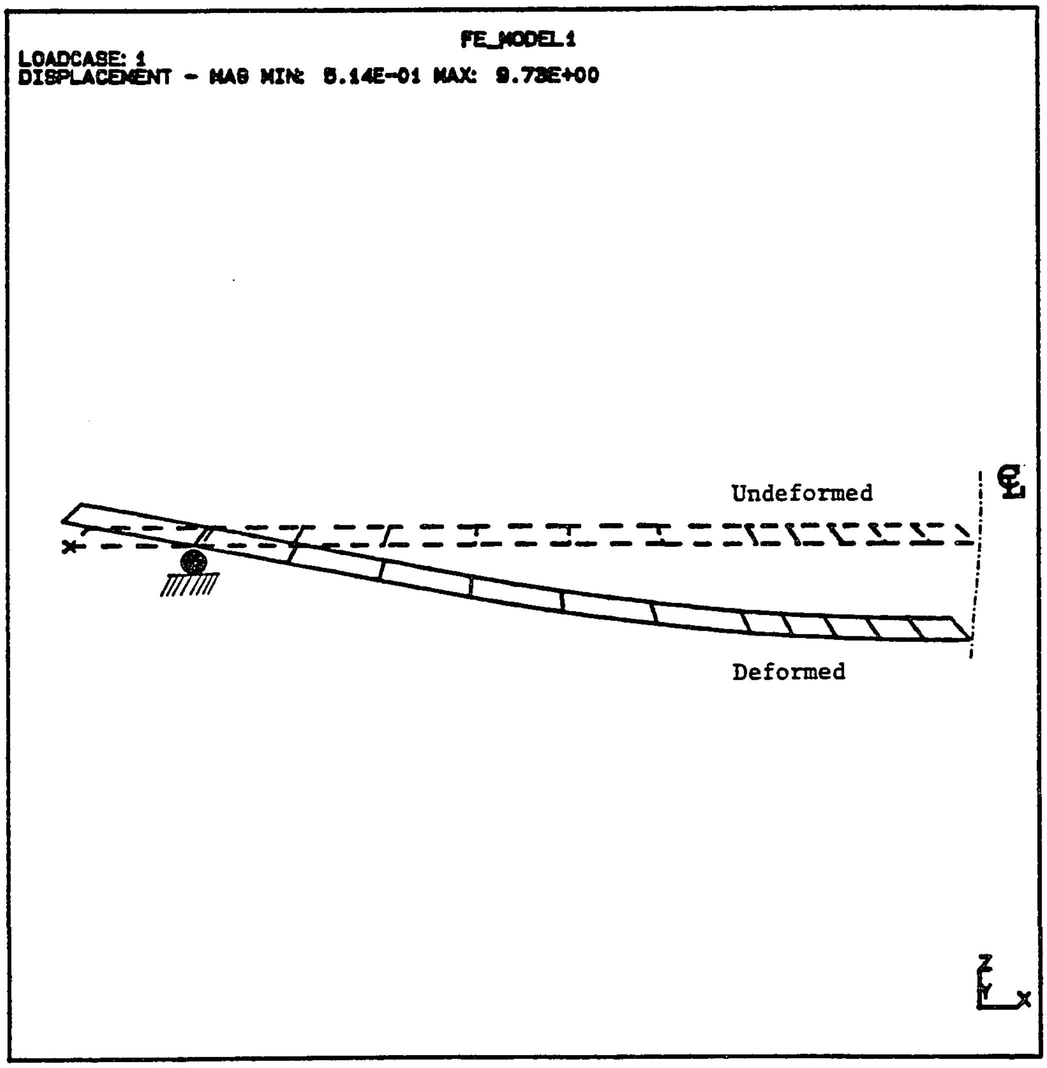


Figure 6.63: Deflection Profile, Specimen Delft Beam, Implicit Layering.

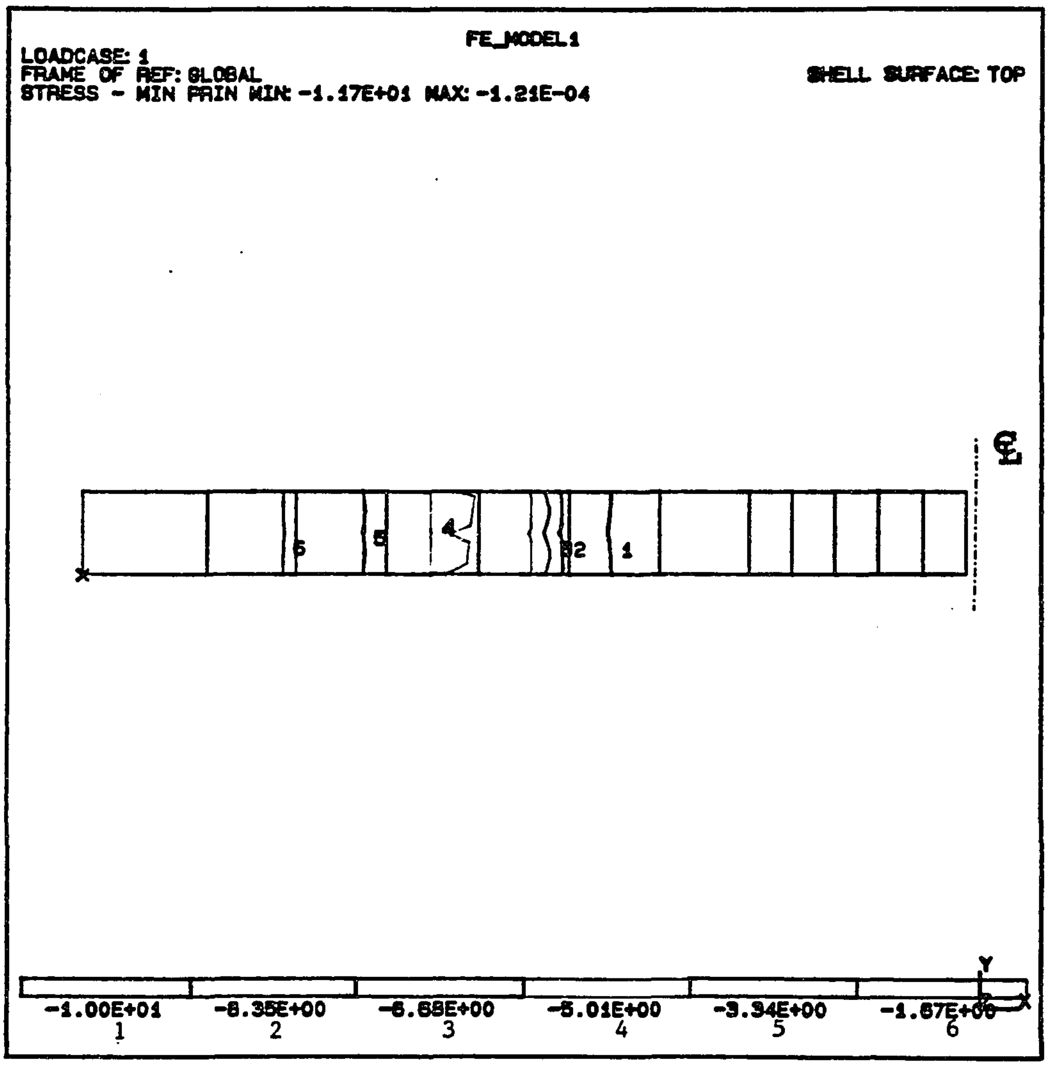


Figure 6.64: Concrete Compressive Stresses-Top Surface, Specimen Delft Beam, Implicit Layering.

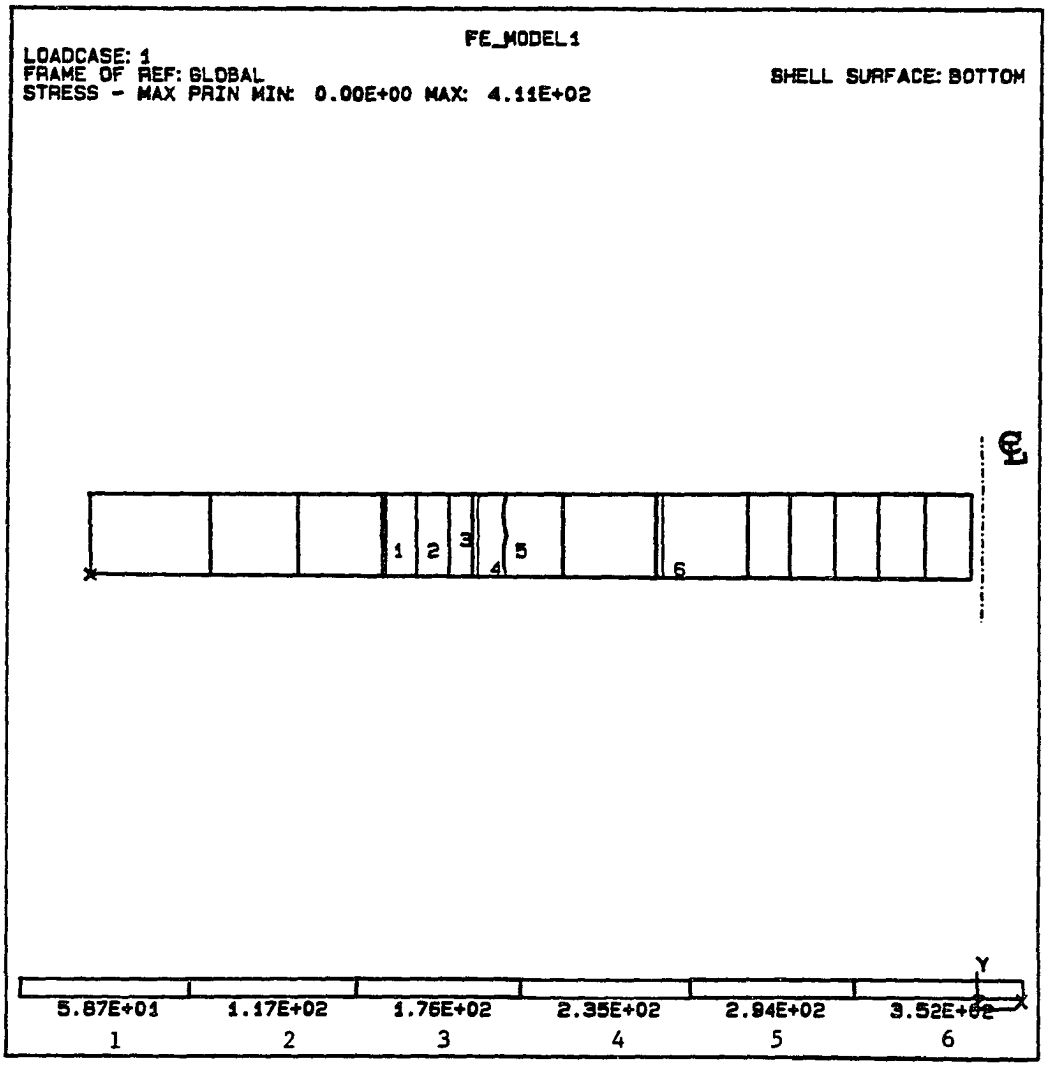


Figure 6.65: Reinforcement Stresses, Specimen Delft Beam, Implicit Layering.

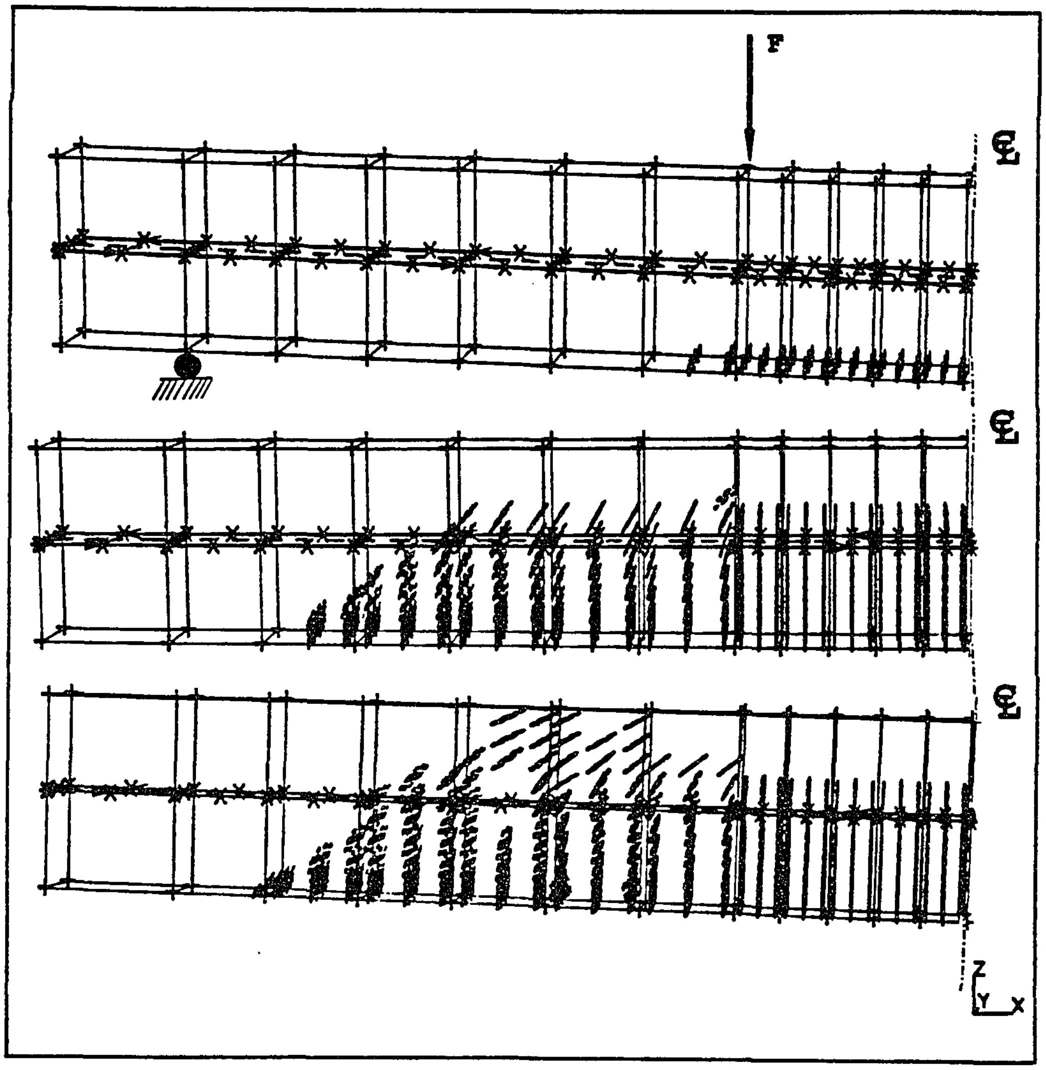


Figure 6.66: Cracking Pattern, Specimen Delft Beam, Implicit Layering.

The deflected shape obtained in the analysis using explicit layering is shown in figure 6.67. The compressive stress distribution on the top surface of the beam is shown in figurefig:cl52c. It is interesting to see that the stress distribution in the flexural zone is not uniform leading to localized crushing under the loads. The stress distribution in the reinforcement is shown in figure 6.69. The crack patterns from the explicitly layered analysis is shown in figure 6.70. These cracks initially develop in the pure moment zone but on subsequent loading some inclined cracks develop in the region between the support and the load. Some slightly inclined cracks appear at a few points in the pure moment zone. A possible explanation for this could be the relatively coarse mesh employed between the loading point and the center (two elements along the length). Thus some boundary effects could cause these shear stresses, especially after cracking. The point directly under the load indicates excessive cracking due to very high localized shear stresses. The load-deflection response from this analysis is shown in figure 6.60. Two different analysis were conducted using different amounts of fracture energy. The smaller amount of 60 N/m resulted in premature crushing. But on increasing the fracture energy to 400 N/m a slightly stiffer response was observed leading to localized crushing failure at nearly 80 KN. This clearly indicates that the explicit layering procedure is much more sensitive to the employed material parameters employed then the implicit layering procedure. In the present study, only implicit layering procedures was used for analyzing flexural test specimen.

Torsional Element ML9

In general reinforced concrete slab elements are subjected to flexural and torsional loads in the reinforcing directions. In lightly reinforced flexural elements a yield line analysis approach is able to predict the ultimate load with reasonable accuracy. In structural elements with large amounts of reinforcement (1.5% or more) such an analysis procedure results in unconservative estimates of the ultimate capacity. The reason for this is that crushing in concrete precedes the yielding in the reinforcement. As discussed in the preceeding sections, diagonally compressed concrete elements undergo considerable softening due to cracking (Vecchio and Collins 1982, 1986). In structural elements that are heavily reinforced, even the lower bound analytical

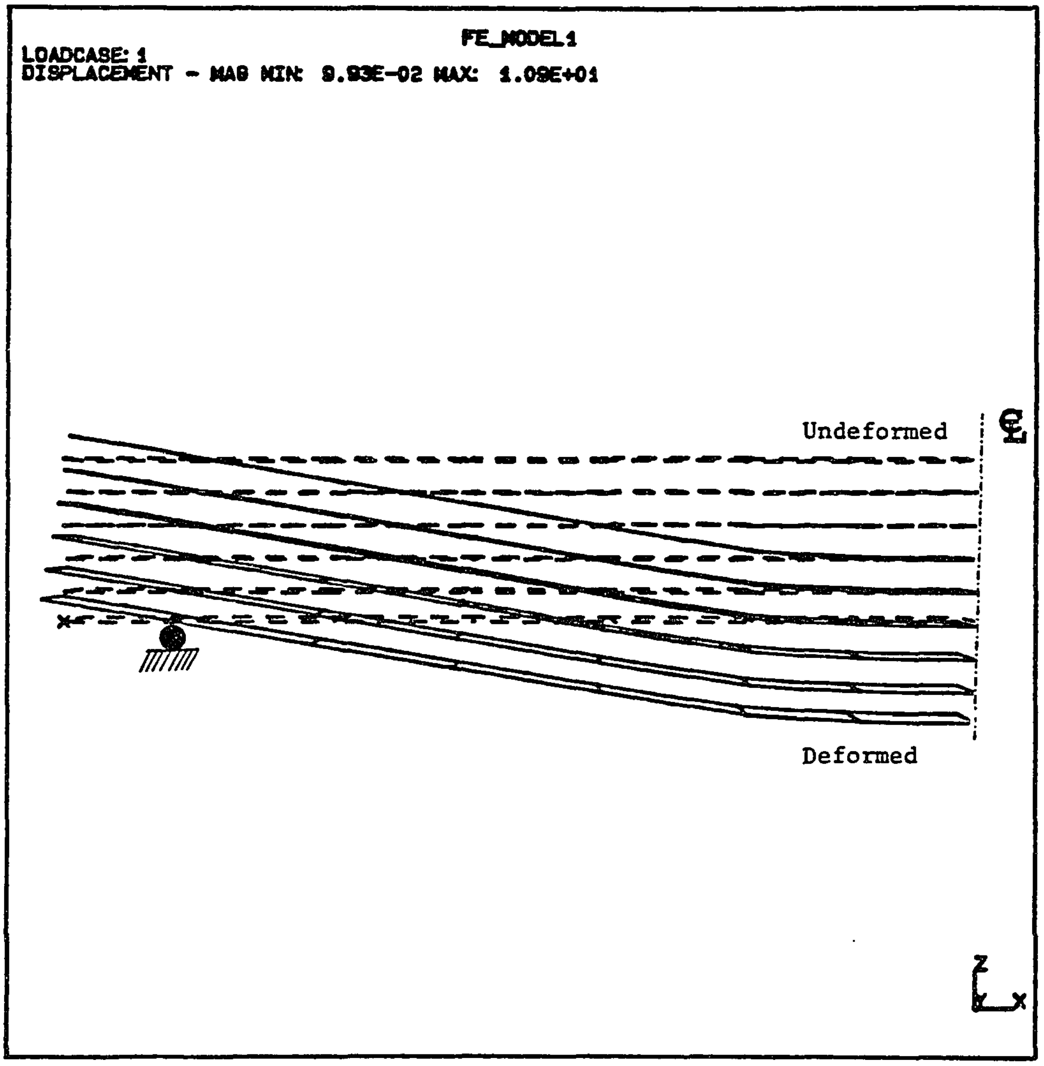


Figure 6.67: Deflection Profile, Specimen Delft Beam, Explicit Layering.

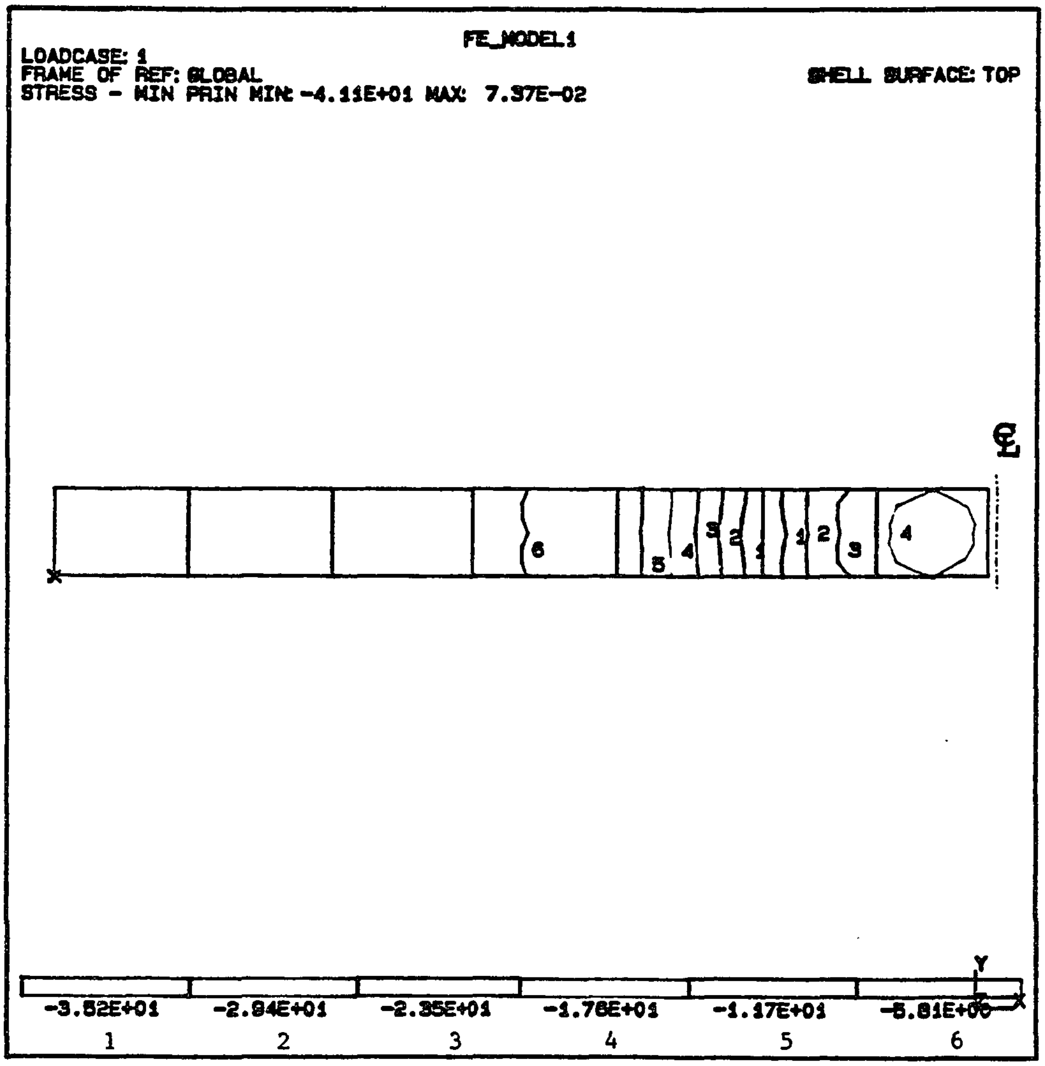


Figure 6.68: Concrete Compressive Stresses-Top Surface, Specimen Delft Beam, Explicit Layering.

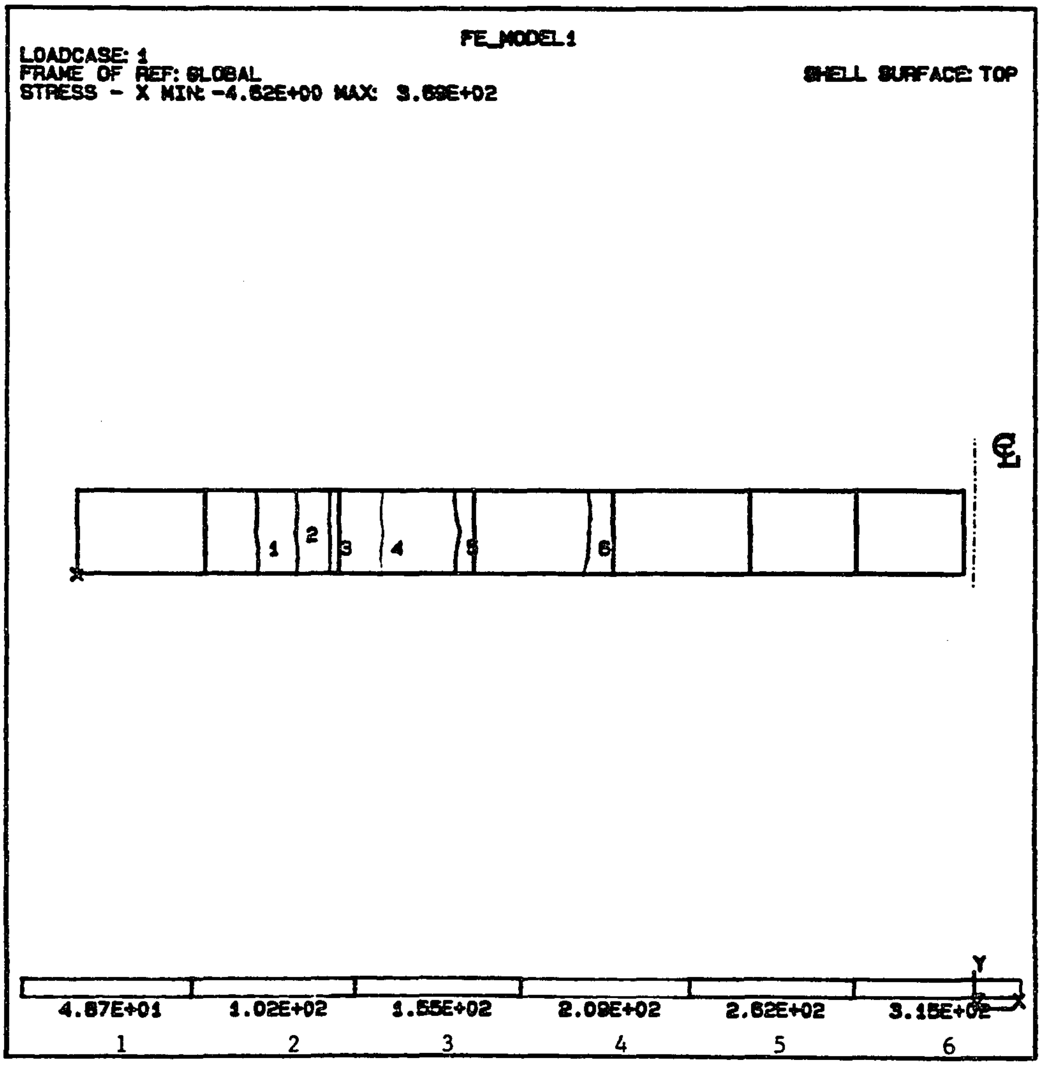


Figure 6.69: Reinforcement Stresses, Specimen Delft Beam, Explicit Layering.

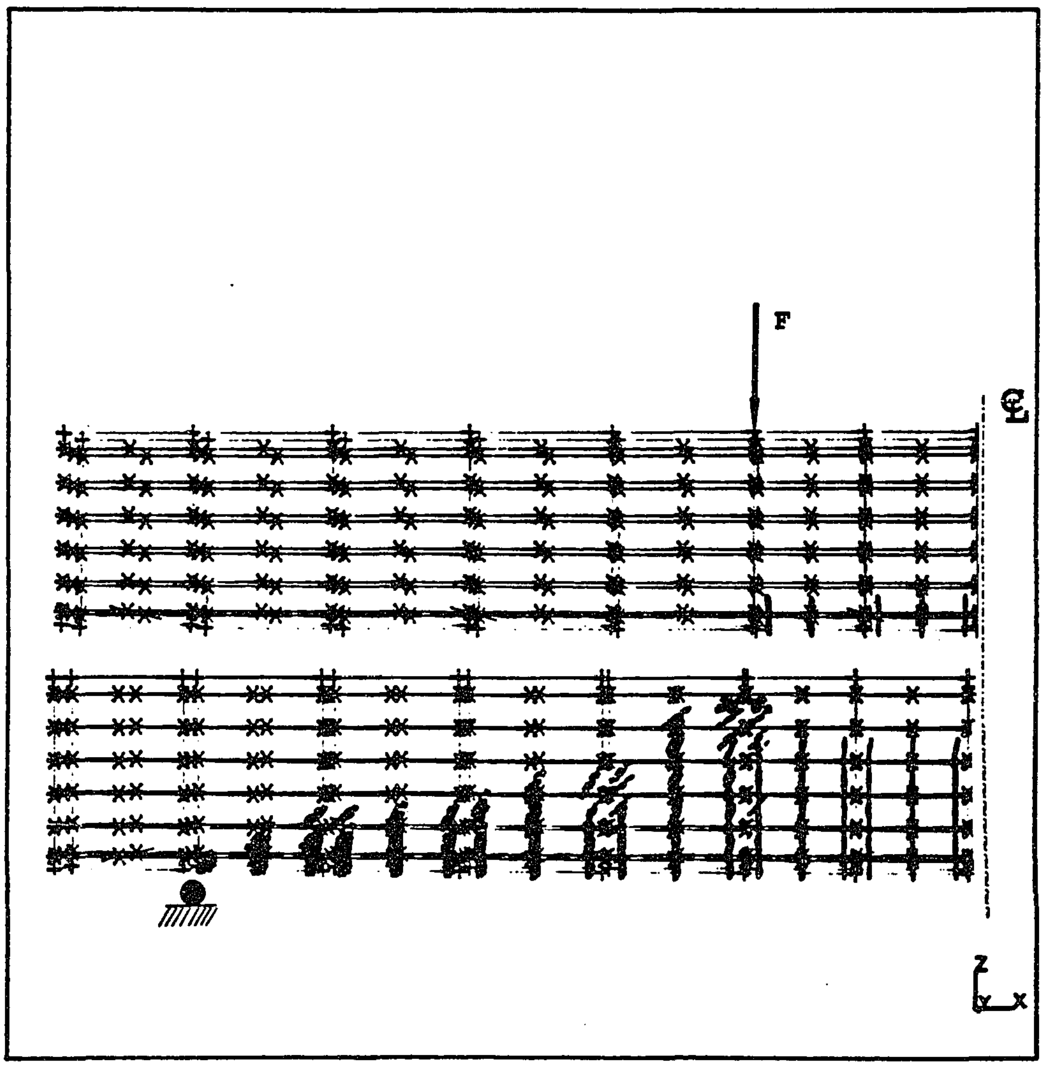


Figure 6.70: Cracking Pattern, Specimen Delft Beam, Explicit Layering.

approaches are questionable as they do not account for these effects. Marti et al. 1987a, 1987b, tested a number of heavily reinforced slab elements (nearly 2%) under torsional loads. In this study one specimen from this series (ML9) was selected and analysed.

The material parameters of the specimen is shown in table 6.12. The slab is a square of sides 1600mm and 200mm thick, having a reinforced mesh on the top and bottom with a cover of 20mm. The torsional moments generated in the experiment by applying equal and opposite loads at adjacent corners were simulated analytically by applying uniform edge moments along the line joining the mid points of the slab sides. Accordingly, the isotropic reinforcement mesh was rotated 45^o (from its original 0^o - 90^o orientation). A single element was used to describe the specimen since the stress field is uniform. The implicit layering procedure was employed to model the cross-section with concrete and steel layers at appropriate positions. A total of fourteen layers were used to discretize the section, of which four layers represent the steel with appropriate orientations (45^o and 135^o to the loading), while the remaining layers were used to describe the concrete. A 3x3x1 Gaussian numerical integration procedure was used to compute the stiffnesses of both concrete and steel layers. After cracking, tension stiffening was considered in only concrete layers adjacent to the steel meshes (two orthogonally placed adjacent steel layers) and were four layers in all. The thickness of the tension-stiffening layers is 20mm each. The remaining concrete layers were assumed to be strain-softening layers after cracking. Thus each concrete layer was approximately 20mm thick. A threshold angle of 15^o for the initiation of new cracks, with respect to previously existing cracks, was employed in the analysis.

The analytical prediction of the moment vs. the principal curvature for this specimen is shown in figure 6.71 along with the experimental results. The analytical prediction is reasonably close to the experimental results. The sudden increase in the curvature seen in the analytical response at a moment of 40 KN-m/m is due to cracks propagating through the thickness of the slab element rapidly. The experimental results were not available at such small load increments. The mode of failure of this specimen was due to crushing of concrete on both slab faces and

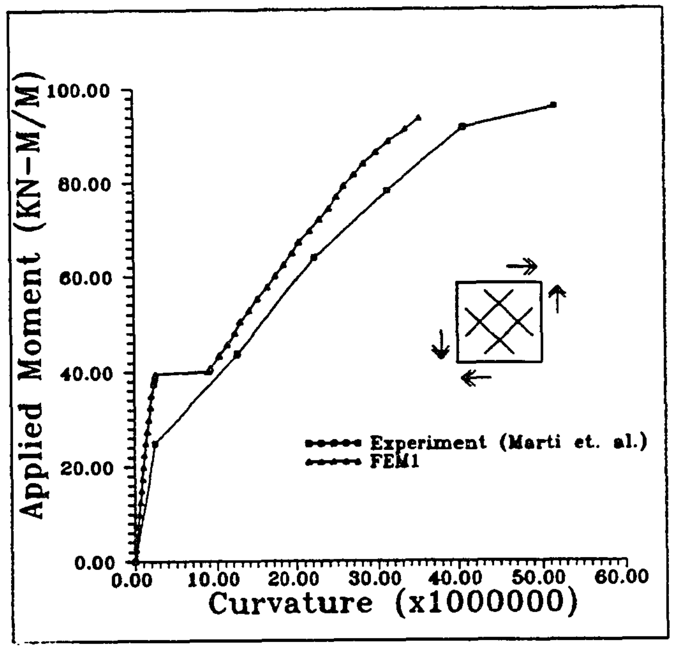


Figure 6.71: Moment vs. Principal Curvature, Specimen ML9

not due to yielding of steel, which is also observed in the analysis. The analysis predicted cracks penetrating to the central two layers (7 and 8) from each face of the slab that are mutually orthogonal. Thus, layers 7 and 8 cracked orthogonally and are in a state of biaxial tension. The neutral axis in each principal direction shifted to the level of the fifth layer from the compression face from its initial position at the center of the section. The load carrying capacity of the slab after cracking was mobilized in the outer four layers (1-4 and 11-14) in each principal direction.The ultimate moment obtained in the present study was 95 KN m/m applied along the edge of the slab versus an experimental value of 104.8 KN m/m. Marti et al. 1987a and 1987b, have also reported the ultimate values obtained from yield line analysis (136 KN m/m), lower bound (114 KN m/m), Compression field theory (Collins and Mitchell 1980, 92KN m/m) and ACI (46.4 KN m/m). It appears that the ACI code predicts an extremely conservative estimate of the ultimate load, while estimates based on either lower bound or yield line analysis are unconservative. The results based on the compression field theory (Collins and Mitchell 1980) and those obtained in the present study account for the compressive strength and stiffness reduction after cracking and are able to predict reasonable values of the ultimate load.

Dudeck Slab S1

RC Slab S1 was an isotopically reinforced slab tested by Dudeck et al. (1978). This slab was supported only at the corners where the transverse deflection was restrained and loaded at the center by means of a concentrated load. Taking advantage of symmetry , one quarter of the slab was analyzed in this study. The dimensions, finite element mesh discretization and the details of the implicit cross-sectional layering procedure is presented in figure 6.72 and 6.73 respectively. The stiffness of concrete and steel layer was computed using a 3x3x1 Gaussian integration scheme. The material properties used in analyzing this specimen are given in table 6.12. A threshold angle of 30°, between any two cracks at the same point was employed for the analysis.

The load-deflection response obtained in the present study is shown in figure 6.74 along with the experimentally obtained results. Figure 6.74 also shows the analytical results obtained by Abdel-Rehman (1981) and Milford and Schnobrich (1984). Abdel Rehman (1981) employed a fixed crack model with a 30° threshold angle between cracks, while Milford and Schnobrich (1984) employed a rotating crack model in their study. The load-deflection response obtained in the present analysis is close to the experimental results up to yielding in the reinforcement. The analysis detected some localized crushing, resulting large number of iterations for convergence. Thus, the analysis was terminated at this stage (53KN- 'ultimate load'). The ductility observed in the experimental results and other analytical investigations may also be due to strain hardening in the reinforcement. The deflected shape of the specimen is shown in figure 6.75. The crack patterns obtained in the present study are shown in figure 6.76 and the yielding of the reinforcement in the bottom layers is shown in figure 6.77and 6.78. The mode of failure obtained in the analysis was by yielding of the tension reinforcement. The ultimate load obtained in the present study is nearly 53KN, while the experimental value is 61.6KN. The failure loads obtained by Abdel Rehman (1981) and Milford and Schnobrich (1984) are

58KN and 60KN, respectively. The ultimate load obtained using yield line analysis is 56KN.

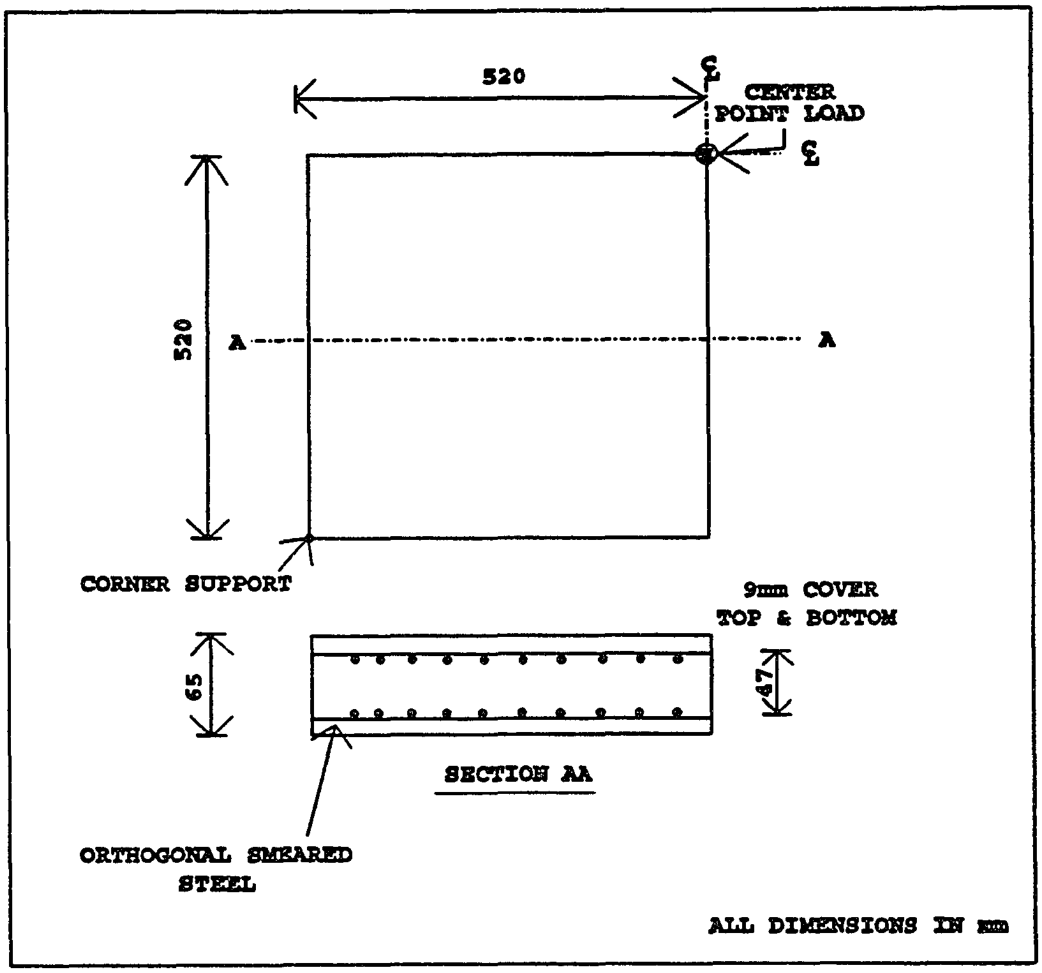


Figure 6.72: Dimensions of Specimen Dudeck Slab S1

Regan Slab series 1#2

This reinforced concrete slab, tested by Regan (1986), was simply supported at the four edges, having a downward load applied at the center by means of a monolithic stud having dimensions 200mmx200mm. In this study the load was applied directly on the slab. The details of the geometry, the implicit layering details and finite

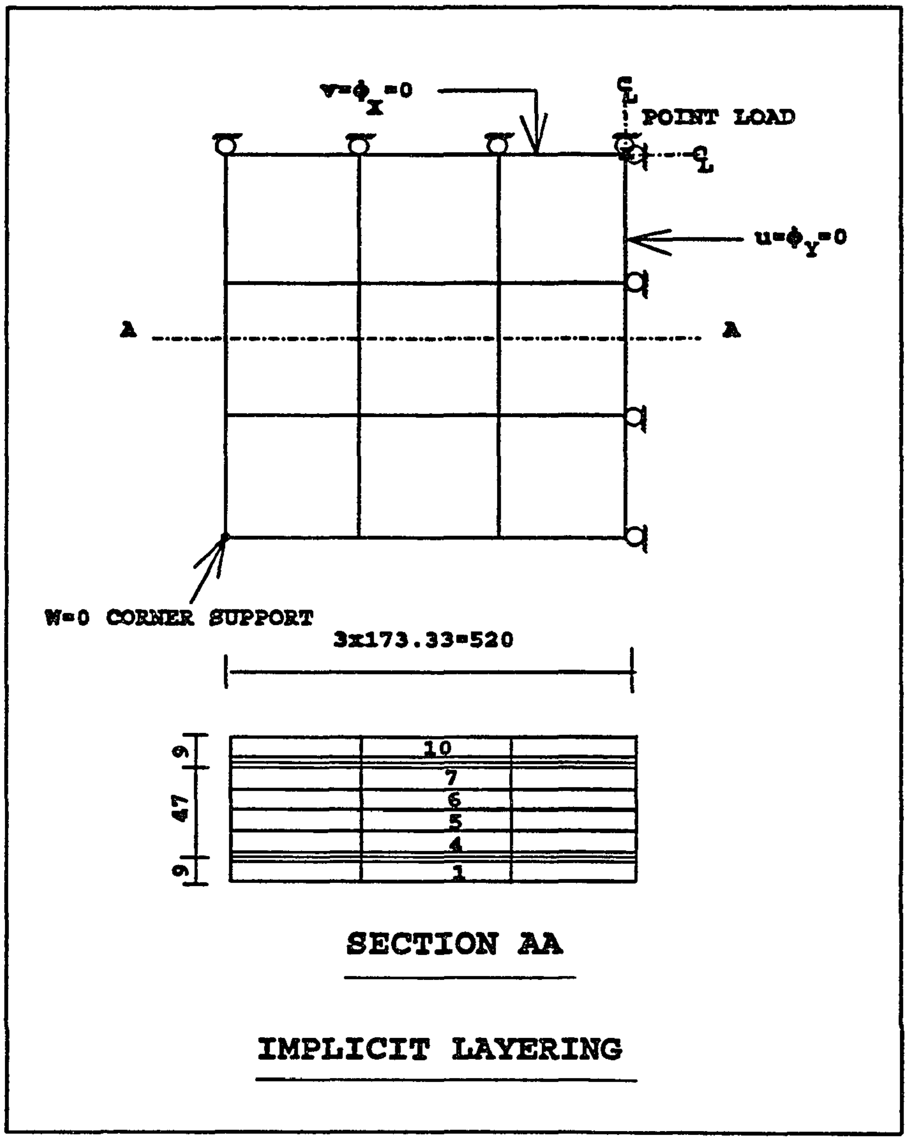


Figure 6.73: F.E. Mesh and Layering Details for Specimen Dudeck Slab S1

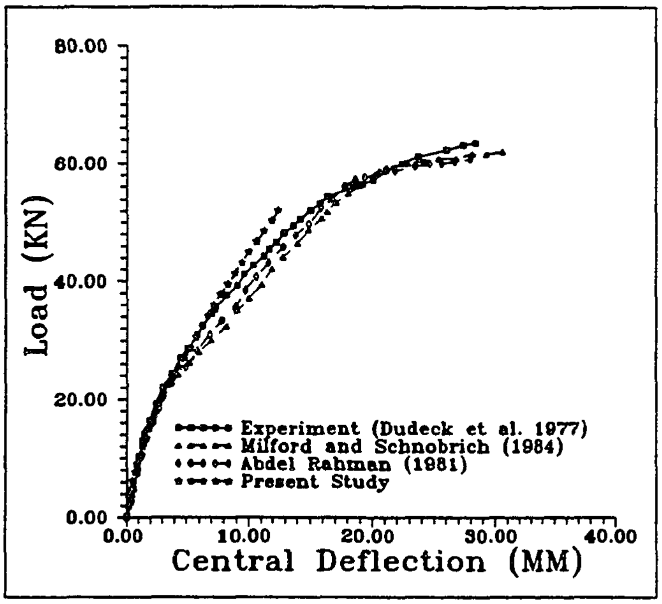


Figure 6.74: Load Vs. Deflection Response, Specimen Dudeck Slab S1

element mesh are shown in figures 6.79 and 6.80. A 3x3x1 Gaussian numerical integration procedure was employed to compute the stiffness contributions of each steel and concrete layer. The material properties are given in table 6.12. The tension-stiffening layer, on each side of the reinforcement, had a thickness of 8mm. A threshold angle of 30° degrees was used in this analysis for the initiation of additional cracks with respect to previously existing cracks at the point. This slab belonged to a series of slabs tested by Regan (1986) to determine the influence of various parameters on 'punching' failures in slabs near the column region. The reinforcement in this particular specimen was uniformly distributed across the slab in the tension zone. The load displacement response for this specimen obtained from the analysis is shown in figure 6.81 along with the experimental results. The mode of failure obtained in the analysis was concrete crushing in the top surface (compression) followed by yielding in the tension steel. No convergence could be obtained at this stage, as seen in figure 6.81. The deflected shape of the specimen is shown in figure 6.82 along with the undeformed shape.

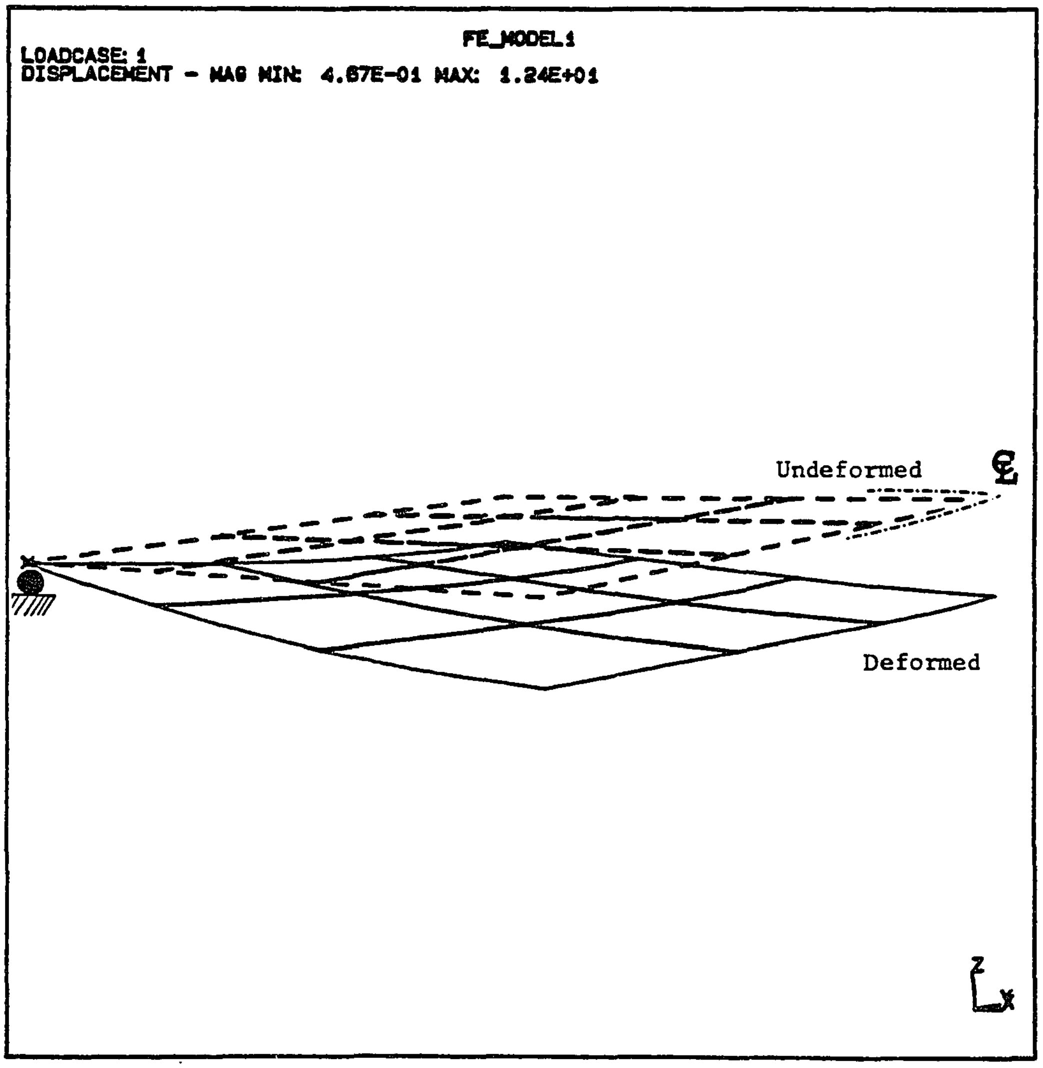


Figure 6.75: Deflection Profile, Specimen Dudeck Slab S1

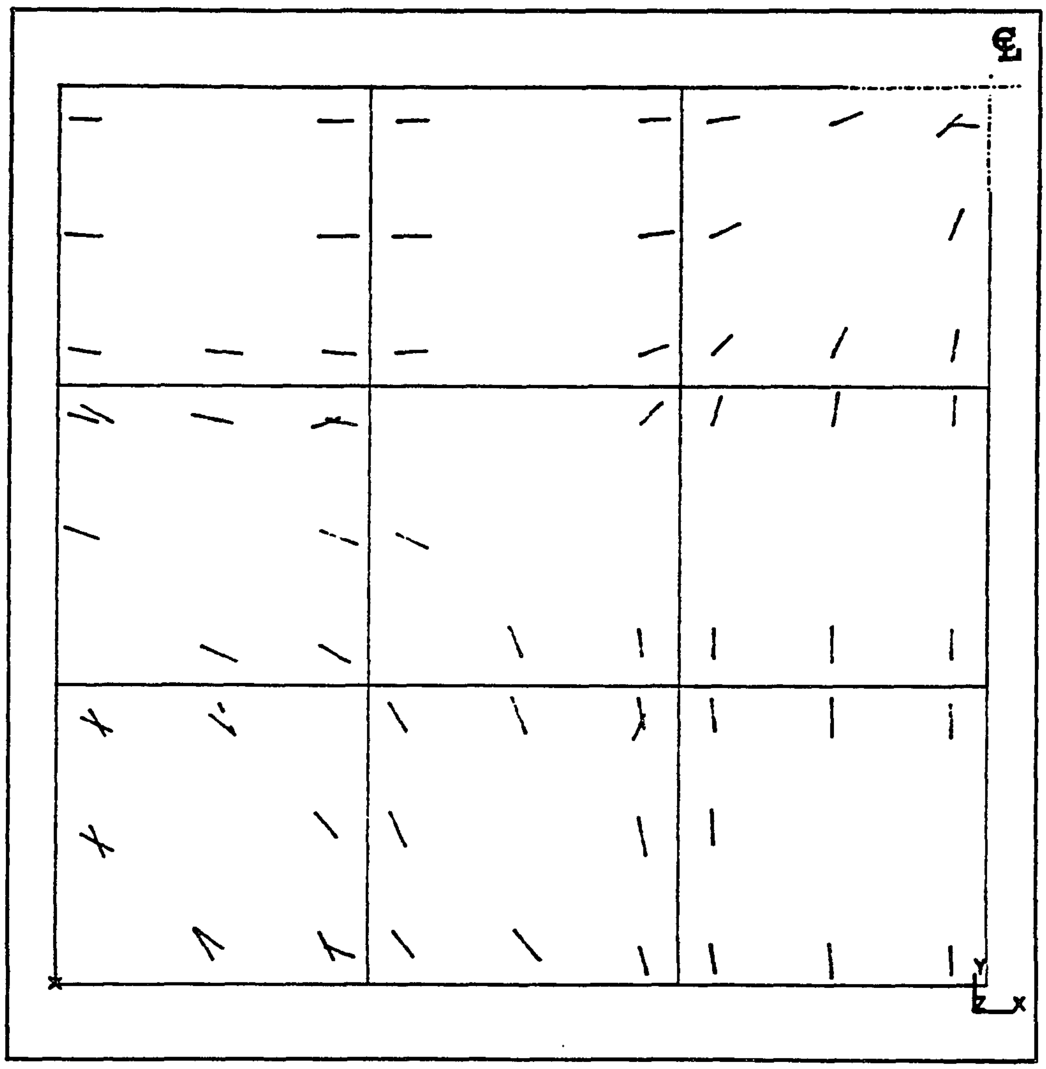


Figure 6.76: Cracking Pattern, Specimen Dudeck Slab S1, Bottom Surface.

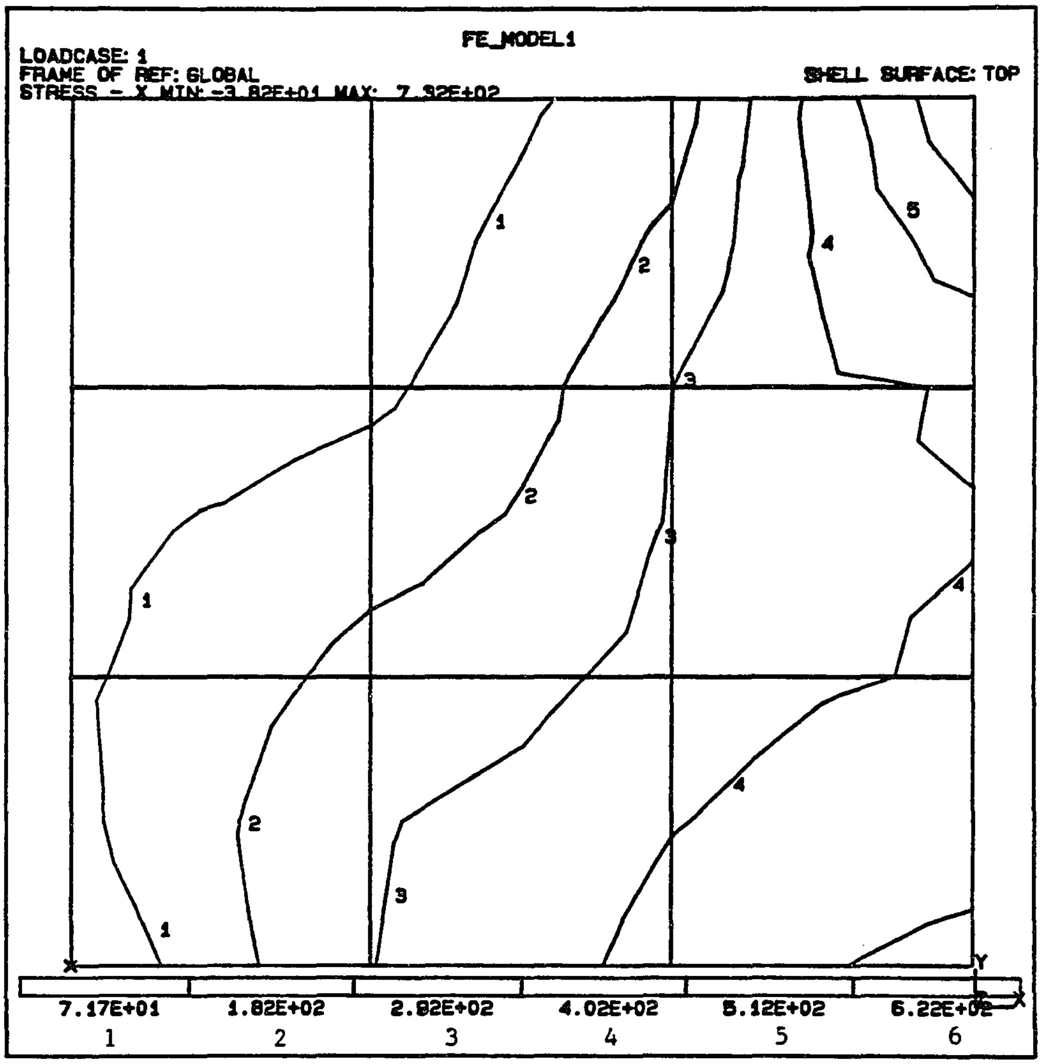


Figure 6.77: Yielding in Bottom Reinforcement, Specimen Dudeck Slab S1, X Direction.

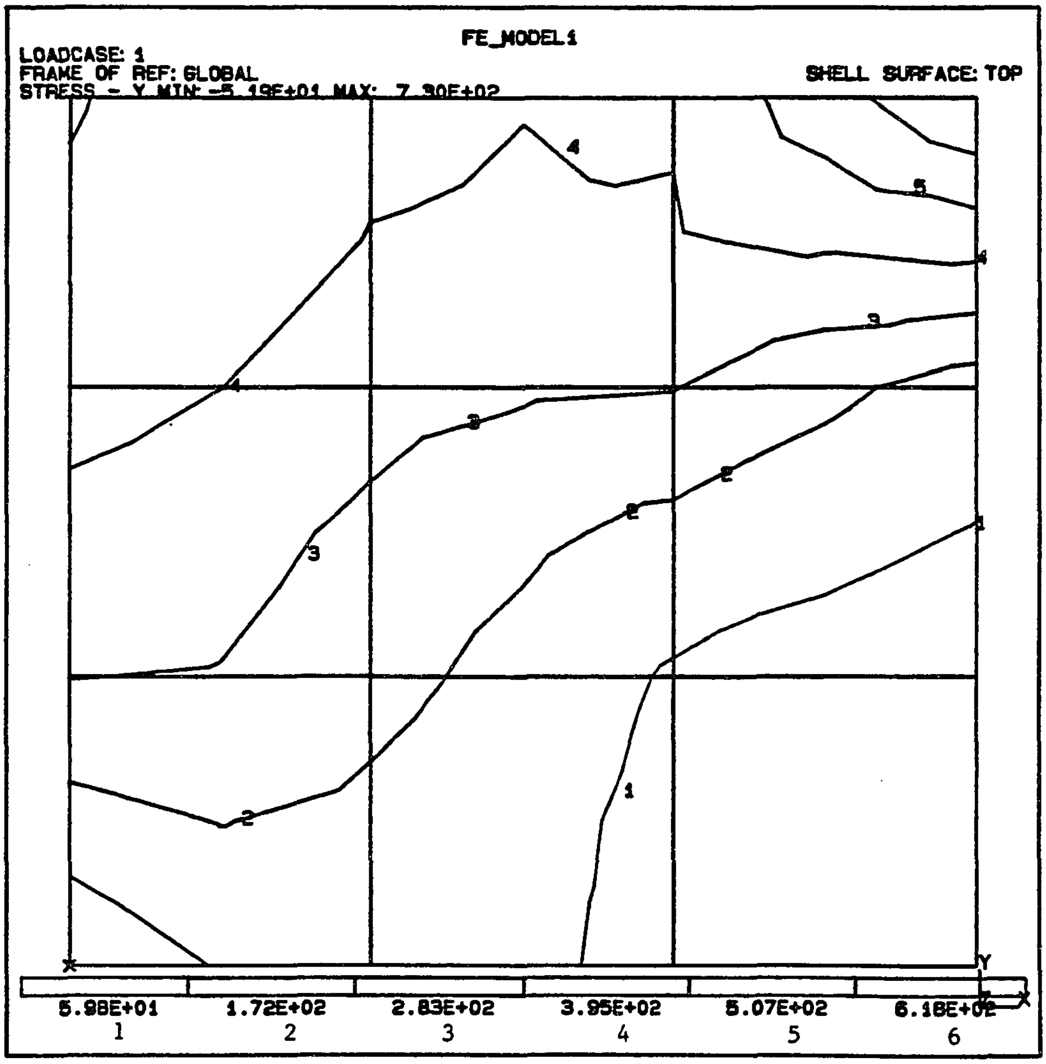


Figure 6.78: Yielding in Bottom Reinforcement, Specimen Dudeck Slab S1, Y Direction.

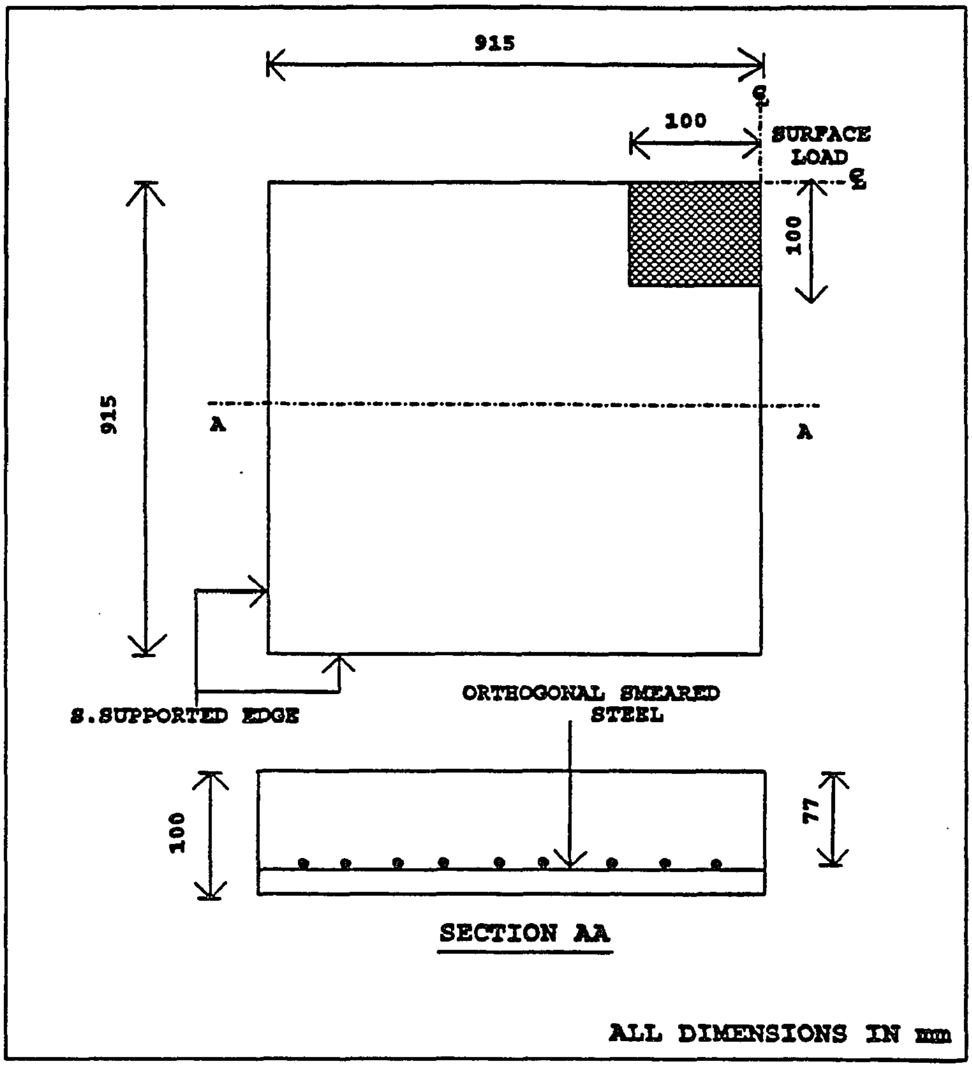


Figure 6.79: Dimensions of Specimen Regan Slab 1#2

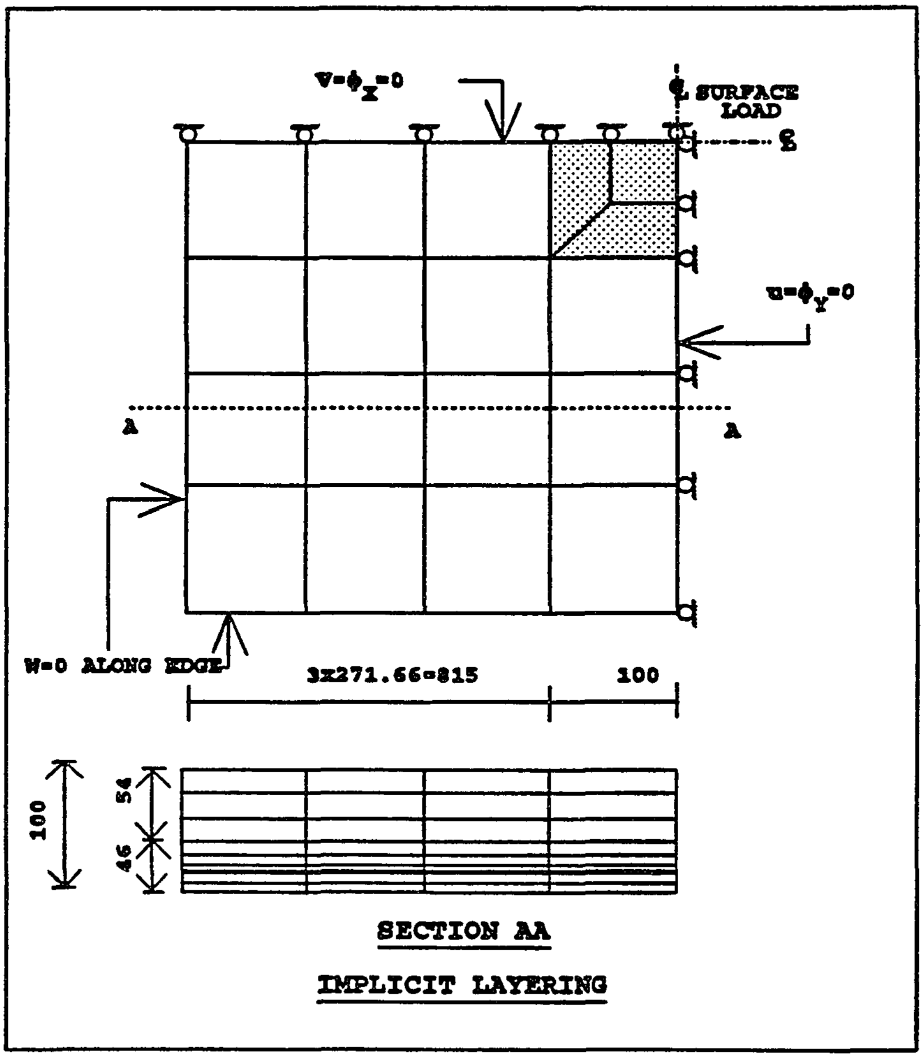


Figure 6.80: F.E. Mesh and Layering Details for Specimen Regan Slab 1#2

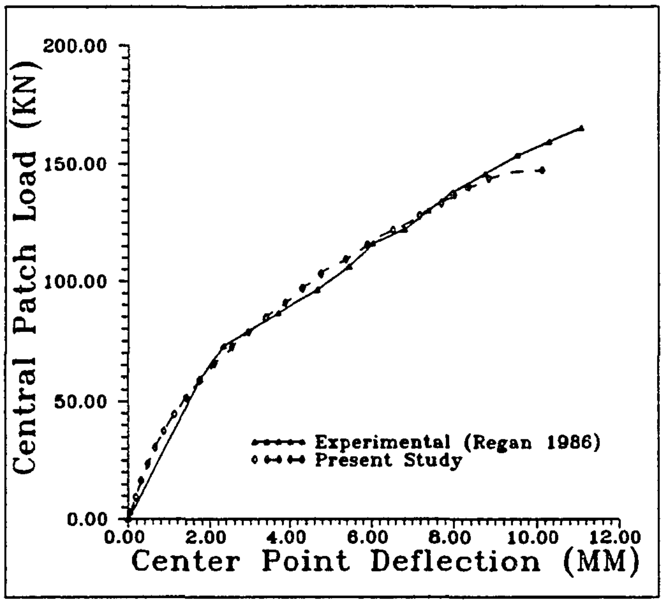


Figure 6.81: Load Vs. Deflection Response, Specimen Regan Slab 1#2

The final cracking pattern is shown in figure 6.83 and 6.84. The cracks near the ultimate load are shown over different cross sections of the slab. The loading in these tests was similar to an inverted column slab connection, resulting in large circumferentially and radially cracked region at the slab bottom surface. The size of these ring cracks are reduced near the mid surface of the slab cross section. This clearly indicates the development of the failure 'cone', typical of the punching mechanism. The cracks appear to reach the mid-surface of the slab. The minimum principal stresses on the compression face of the slab near ultimate is shown in figure 6.85. The stress distribution in the reinforcement is shown in figures 6.86 and 6.87. The implicitly layered analytical solution appears to predict both the load-deflection response and the stress distribution adequately for this slab. In the experiments, Regan (1986) found that the variation in the distribution of the reinforcement had no influence on the load carrying capacity of the slabs. However, the deflection in slabs with reinforcement closer to the loaded area (connection region) was observed to be somewhat less. This indicates that the reinforcement in the 'column strip' of the slab is a better measure of the percentage of reinforcement required in design as compared to using the entire slab width between column centerlines. The ultimate load for this slab obtained in the analytical study was 146 KN (implicit layered analysis). This was approximately 84% of the experimental value of 176 KN. Regan (1986) also reported the predictions obtained from four different codes- British standard 8110, CP110, ACI 318 and CEB-FIP. The critical failure sections considered for this analysis by Regan (1986), as per the different codes, for these prediction is shown in figure 6.88. The failure values obtained from these analyses are 178 KN (BS 8110), 172 KN (CP 110), 138 KN (ACI 318) and 134 KN (CEB-FIP). Yield line analysis of this slab resulted in a failure load of 136 KN as reported by Regan (1986). The values predicted by the British codes (BS 8110 and CP 110) are based on larger failure zones, compared with the ACI and the CEB-FIP codes. Additionally the limiting shear stress at the failure zone is a function of only concrete strength in the ACI code, while the remaining codes include the contribution of the flexural reinforcement and the slab depth. The safety factors built into the respective codes were not considered for the purpose of this comparison. The analytical failure prediction obtained in this study, was about 10% higher than the yield line prediction indicating that the contribution of the compressive strength of concrete, which undergoes some strength reduction due to cracking, is significant. The failure mode appears to be 'punching' rather then yielding of the slab. The ACI and the CEB-FIP predictions are almost equal to that of the yield line analysis prediction. The predictions obtained using the British codes are much closer to the experimental value which could be due to the larger critical failure section considered. The cracking patterns obtained in this study (6.83 and 6.84) indicated that the 'failure surface' is about 135mm from the face of the 'column' (loaded area), which corresponds to a value of 1.35 times the slab thickness (1.35d).

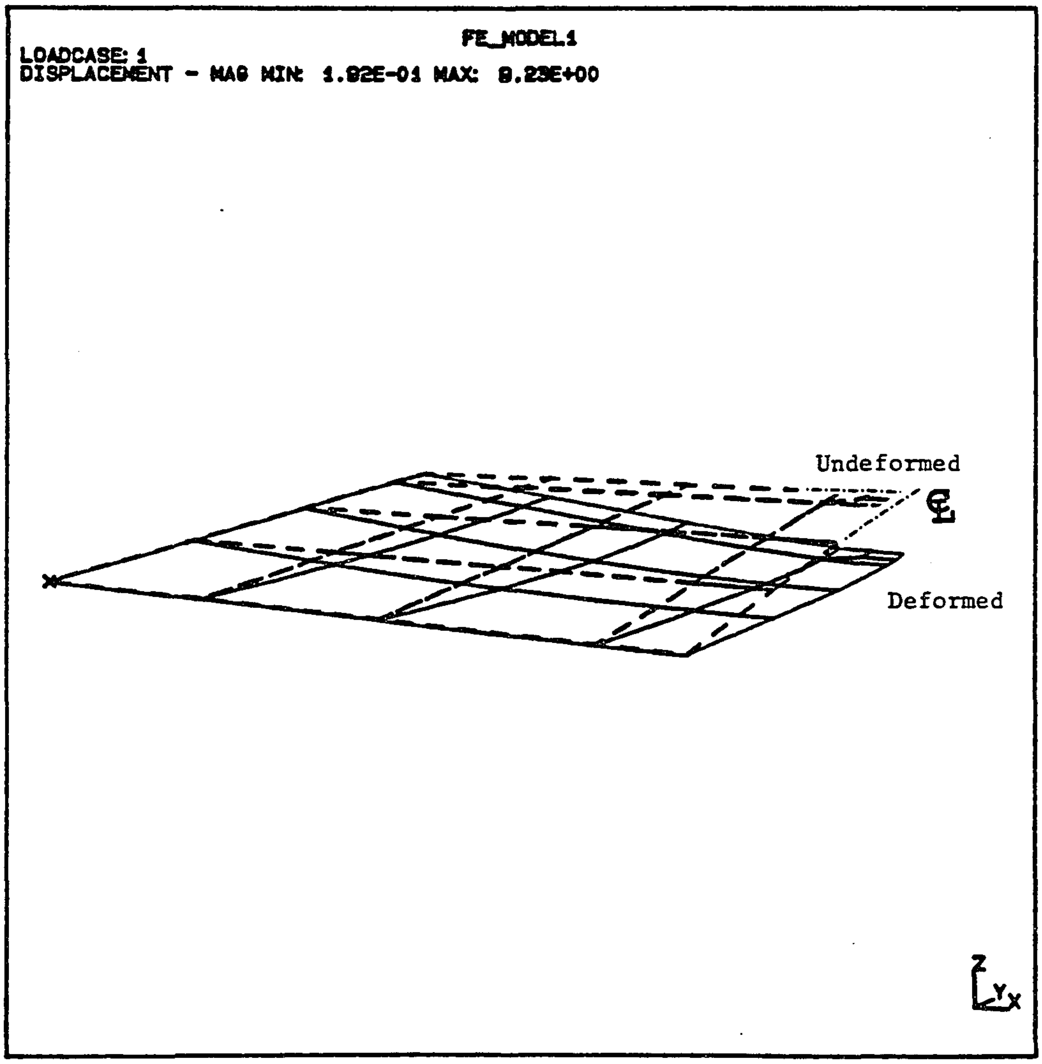


Figure 6.82: Deflection Profile, Specimen Regan Slab 1#2

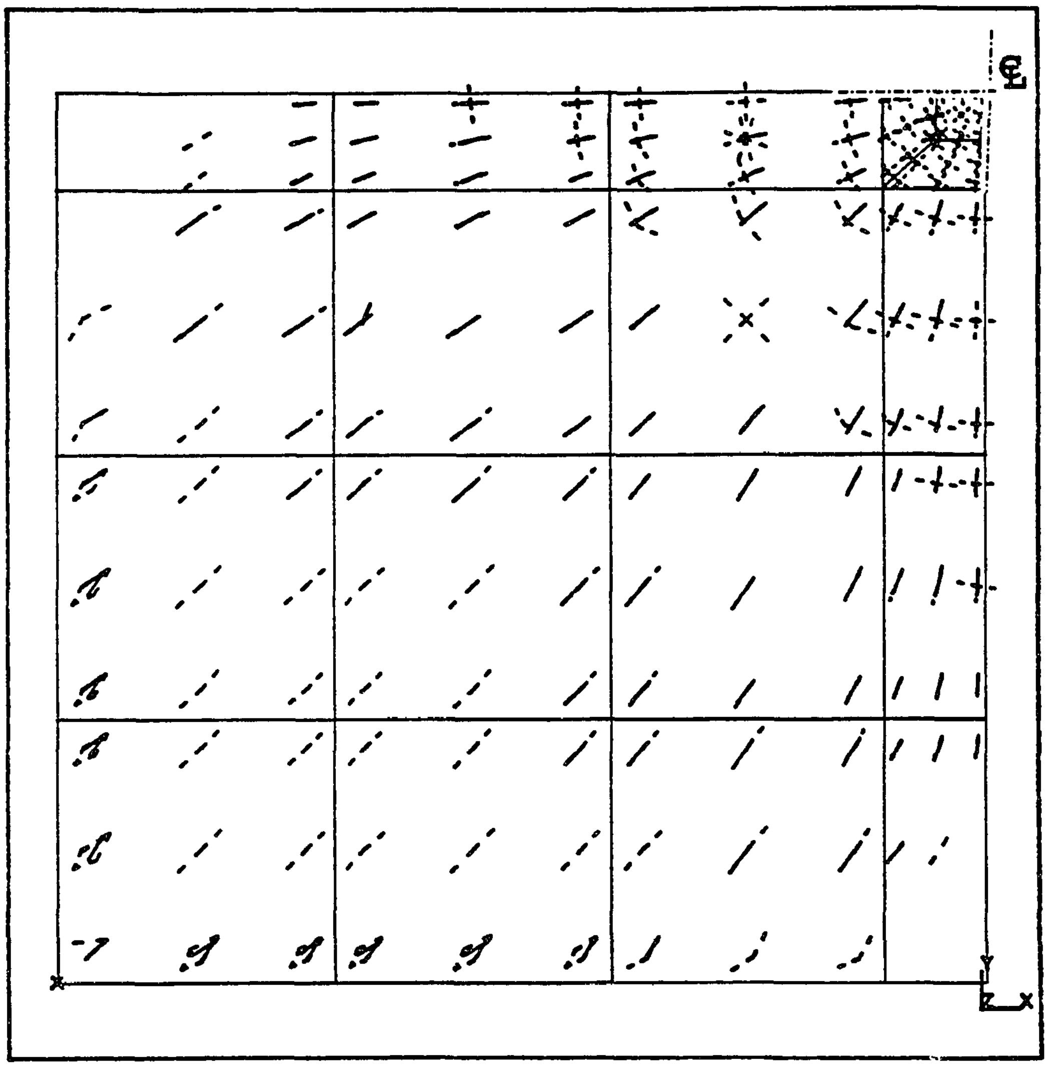


Figure 6.83: Cracking Pattern, Specimen Regan Slab 1#2, Bottom Surface.

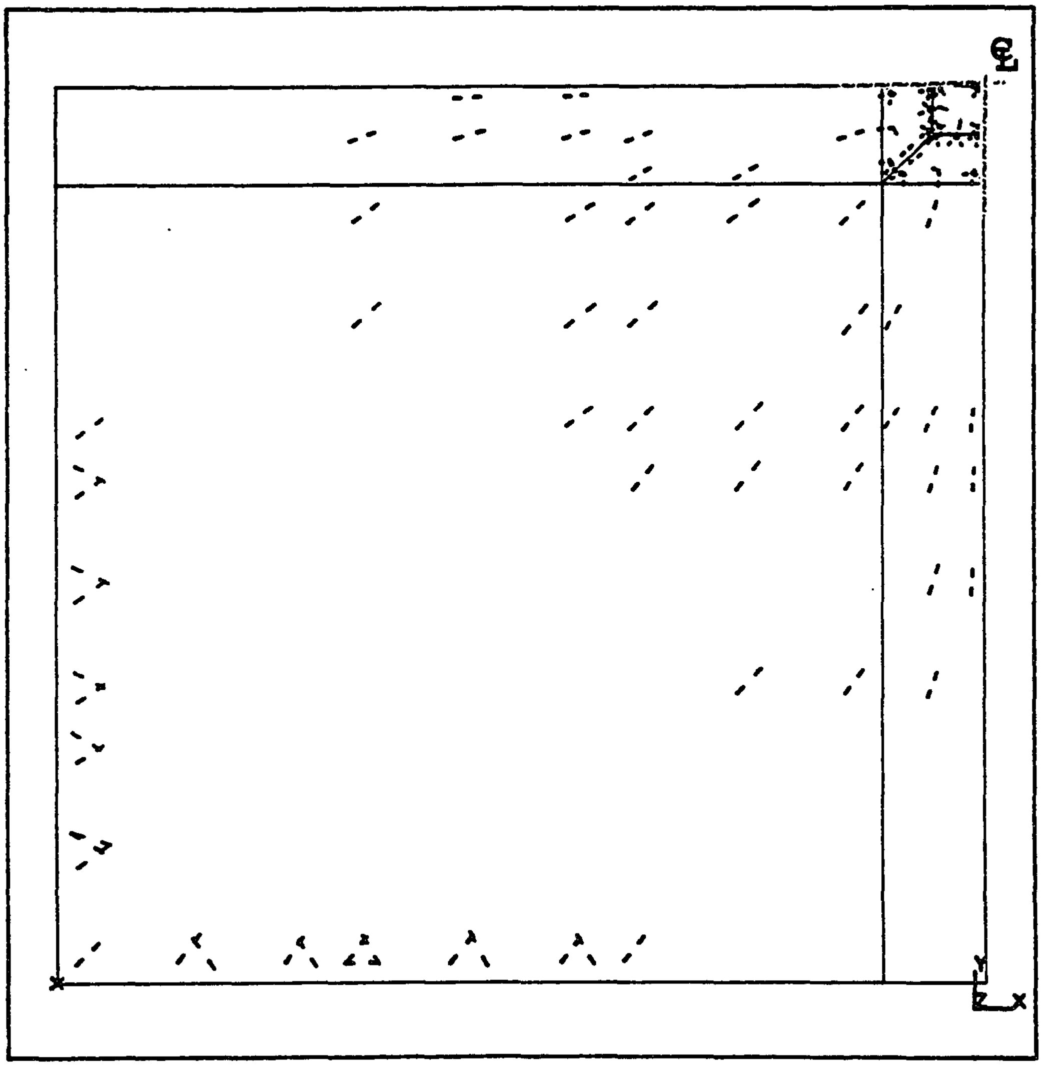


Figure 6.84: Cracking Pattern, Specimen Regan Slab 1#2, Middle Surface.

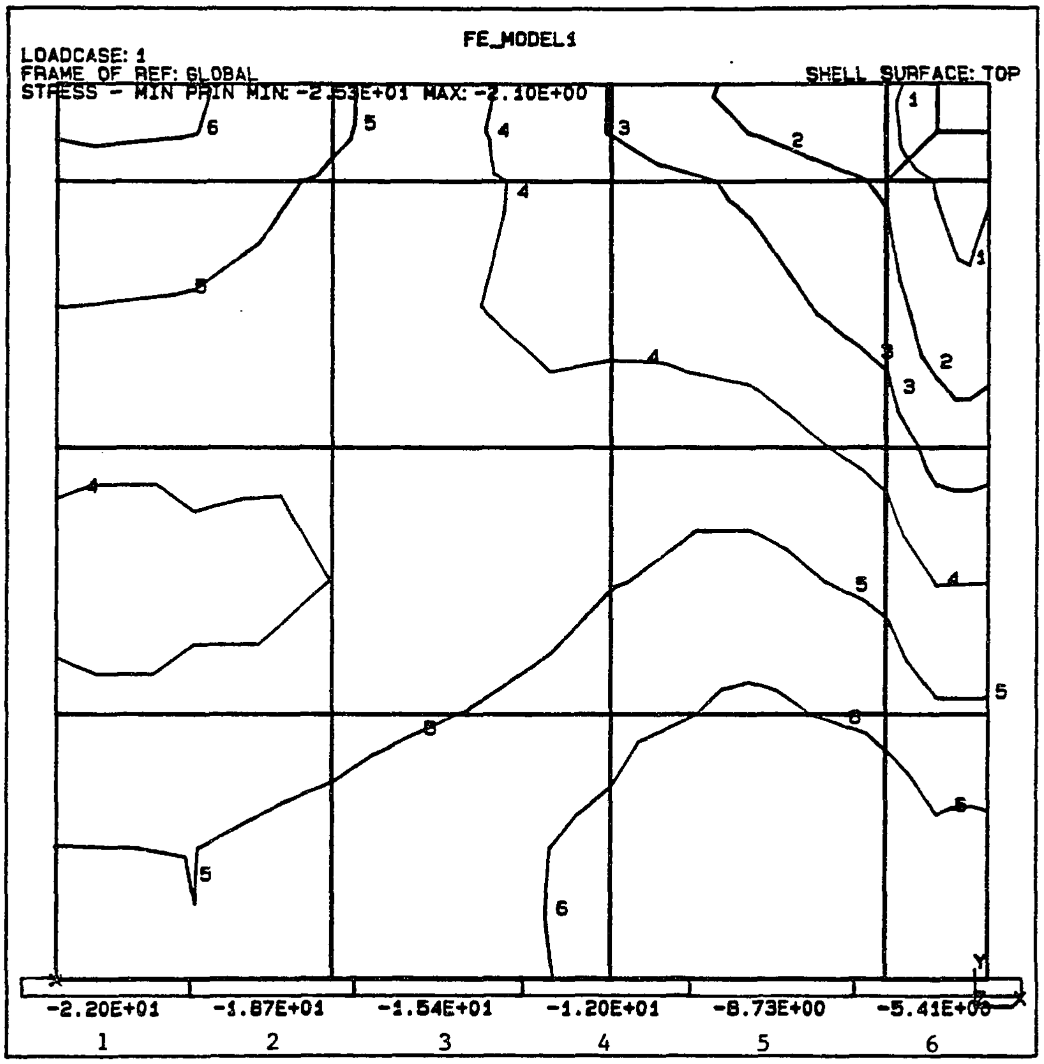


Figure 6.85: Minimum Principal Stress Distribution on the Compression Face, Specimen Regan Slab 1#2, At the Ultimate Load.

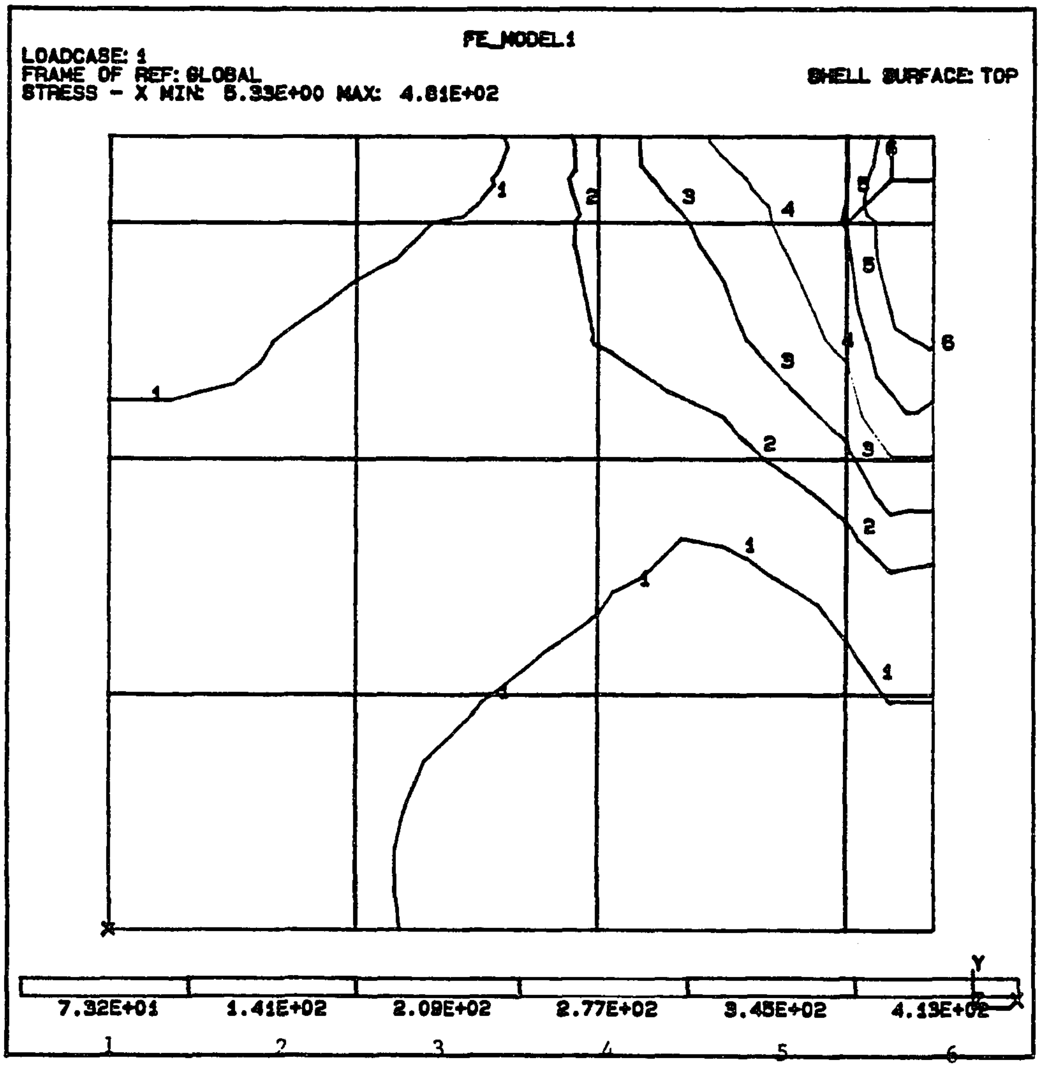


Figure 6.86: Stress Distribution in the Reinforcement, X-Direction, Specimen Regan Slab 1#2

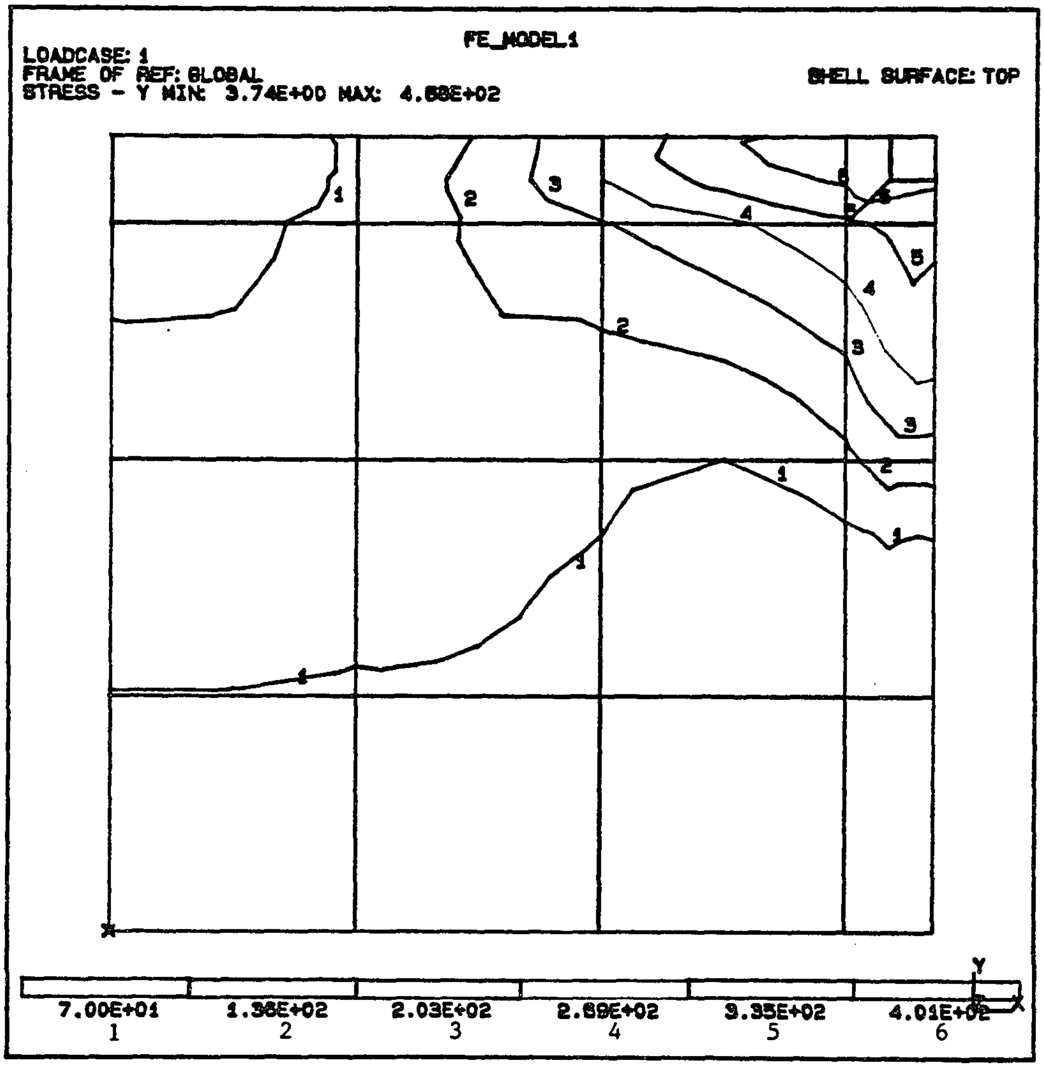


Figure 6.87: Stress Distribution in the Reinforcement, Y-Direction, Specimen Regan Slab 1#2

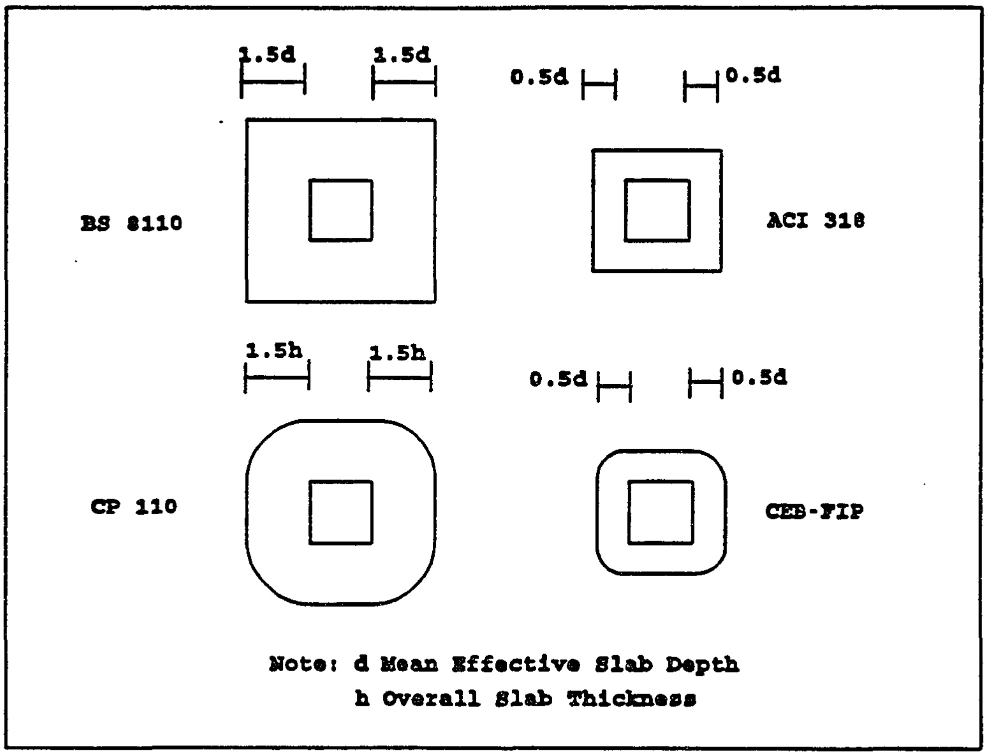


Figure 6.88: Critical Section, Around Columns, Employed in Different Codes.

6.4 RC Slab Element Subjected to General Loading

Slab Elements examined up to this point in this study have been simply supported at the edges and subjected to an axial load in the central region, simulating a column slab connection. Such loadings results in either a ductile load-deflection response and failure due to yielding of the tension reinforcement (eg. Dudeck slab S1, where the reinforcement percentages are low), or a brittle crushing failure at the center of the slab characterized by punching of the 'column' through the slab along with a portion of the slab (eg. Regan slab 1#2 where reinforcement levels are moderately high). While design codes address the need to have design criteria to restrict the possibilities of brittle failures, they do not have provisions to account for the influence of 'lateral restraint' on the punching capacities of slabs, developed due to boundary restraint at the edges. In the experimental study conducted by Regan (1986) a group of slabs that were subjected combinations of uniform edge moments and a central axial load were examined. These slab specimen have a shape that is like a cross in plan, with a 1.73m square central panel and projections of 635mm on each side. The details and other dimensions are provided in figure 6.89. An upward load was applied at the center of the slab by means of a 160mm square plate, while downward line loads were applied at the sides of a 1.83m square. The assembly is supported by means of rollers 457mm beyond the downward line load. By adjusting the ratios of the downward line load and the upward central load various levels of restraining moment were generated along the sides of the 1.83m square. This moment to load ratio was kept constant through the test for each slab, while it was varied between slabs. In this study one specimen from this series was analyzed with the moment per unit width to load ratio ($\frac{m^1}{P}$) of 0.036. One quarter of this specimen has been analyzed using the implicit layering procedure, taking advantage of symmetry. The details of the finite element mesh are provided in figure 6.90. The material properties of the constituents are given in table 6.13. A threshold angle of 30° for the initiation of new cracks with respect to previously existing cracks was employed for this analysis. The central square portion of the slab having an orthogonal mesh reinforcement at both the top and bottom region of the slab

Table 6.13: Material Property of RC Slab Element Subjected to General Loadings

Specimen	Concrete			Reinforcement Mesh			
	f'_c (MPa)	ϵ'_c $\frac{mm}{mm}$	f'_t (MPa)	ρ^{Top} (%)	f_y^{Top} (MPa)	ρ^{Bottom} (%)	f_y^{Bottom} (MPa)
Regan Slab 4#3	34.1	0.0023	2.75	1.3	525	0.64	510

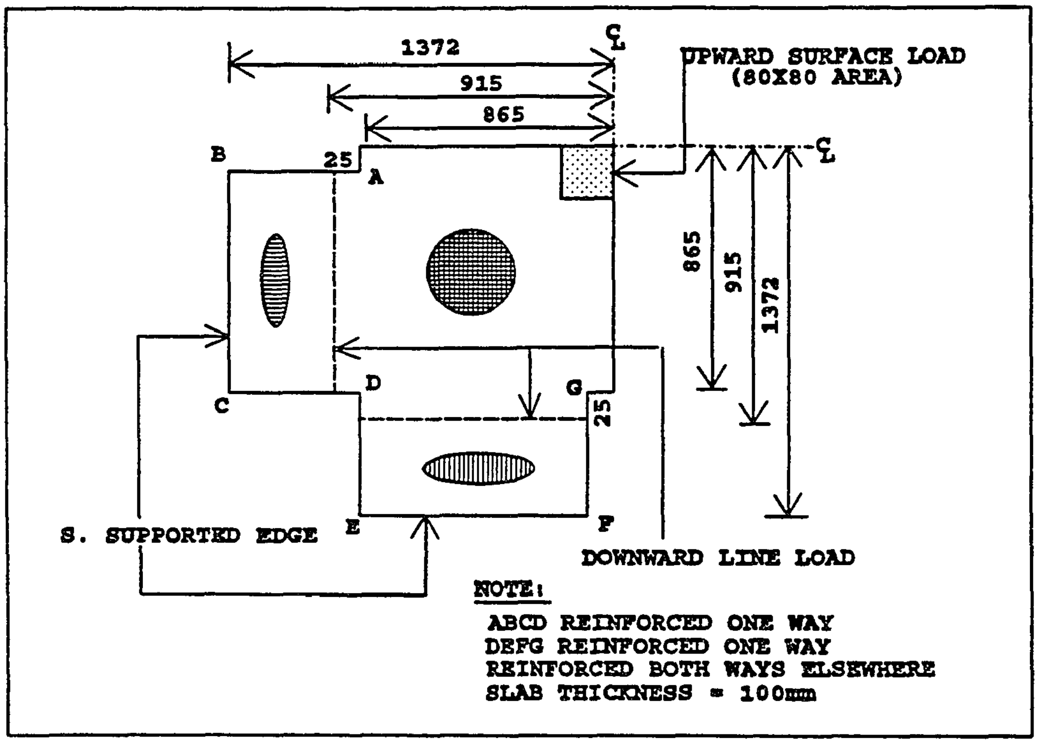


Figure 6.89: Dimensions of Specimen Regan Slab 4#3

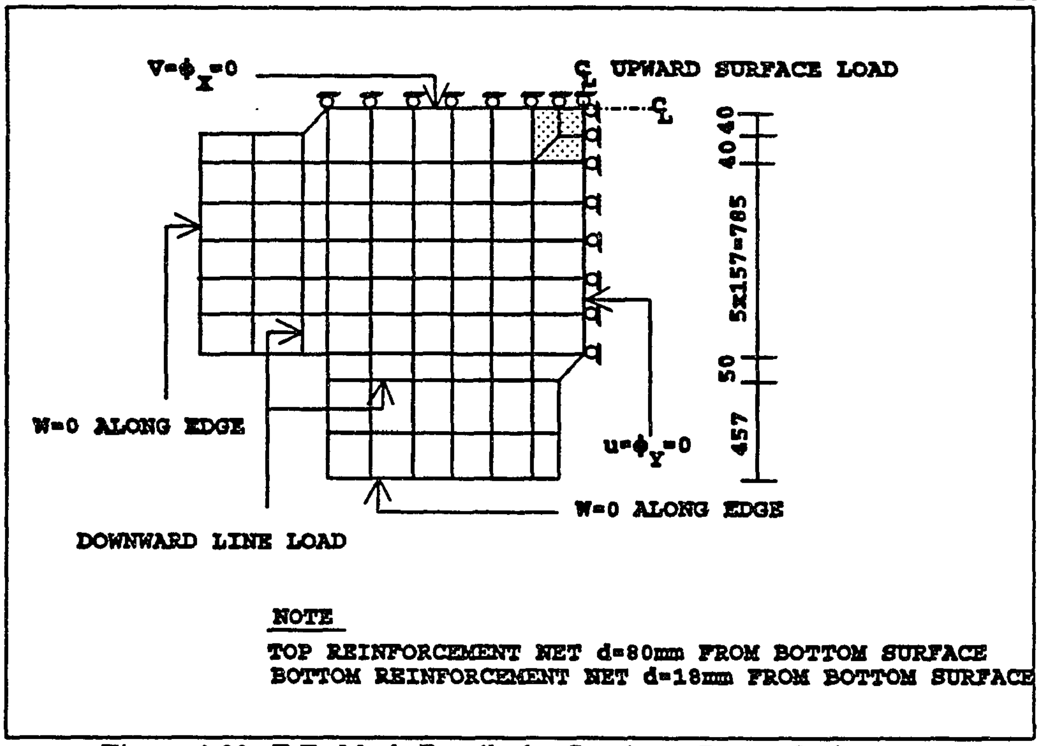


Figure 6.90: F.E. Mesh Details for Specimen Regan Slab 4#3

(cover of 23mm on each face) was represented using ten layers, four steel and six concrete layers. The projections of the slab, marked ABCD and DEFG respectively in figure 6.89, was uni-directionally reinforced at both the top and bottom surface (effective cover of 23mm) respectively. These portions of the slab were represented by eight layers through the cross section, two steel and six concrete layers. The tension-stiffening layer, one on each side of the steel mesh, was 8mm thick.

Figure 6.91 shows the load vs. strain response on the compressive surface of the slab, at a point on the diagonal 95mm from the central loading. The average rotations of the top surface (calculated with respect to the deflection at the center of the slab) with increasing loads is shown in figure 6.92. The deflection profile is shown in figure 6.93. The cracking pattern at 30% of the experimental ultimate load is shown in figure 6.95 and 6.96. The analytical results were obtained up to 50% of the experimentally observed ultimate. However, at this stage the number of cracks initiated increased significantly and the analytical procedure was terminated as it became computationally cost prohibitive. The load strain response and the load rotation response indicate that the analytical model is close to the experimental results. One possible reason for the increased flexibility and excessive cracking observed in the load-strain response in the analysis is a reduced participation of the tension-stiffening response. This may be due to a very thin layer of concrete (8mm) being considered as a tension-stiffening layer on either side of the reinforcement. The crack patterns indicate that due to the upward central load, radial and circumferential cracks are seen in the top surface of the slab near the center. The lower surface has flexural cracks along the line of action of the downward loads. The compressive stresses in the bottom surface of the slab is shown in figure 6.94. Experimental evidence presented by Regan (1986) indicated that even for small ratios of edge moment to punching load, the punching strength of the slab elements increased by as much as 25capacity when no boundary restraints were placed. The analytical model was able to predict the dominant deformational response characteristics up to half the ultimate load.

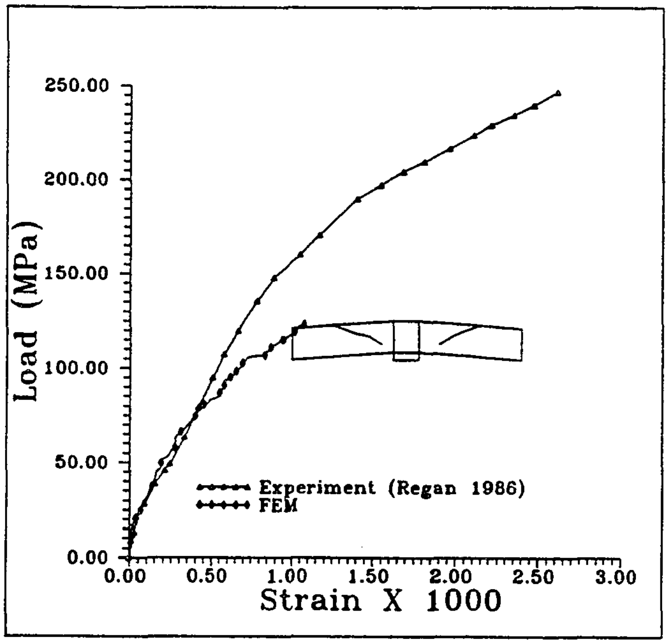


Figure 6.91: Load Vs. Strain Response, Specimen Regan Slab 4#3, At Corner of Central Load.

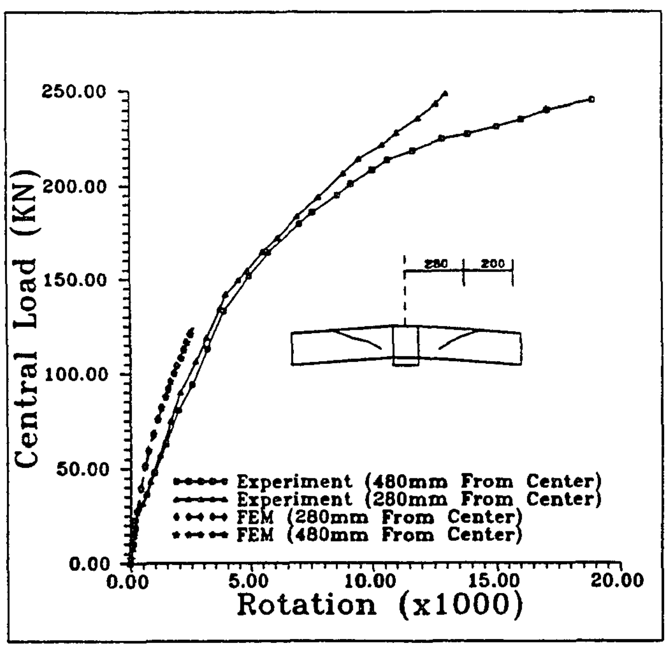


Figure 6.92: Load vs. Rotation Response, Specimen Regan Slab 4#3, At Point Outside Central Load (x=280mm and x=480mm)

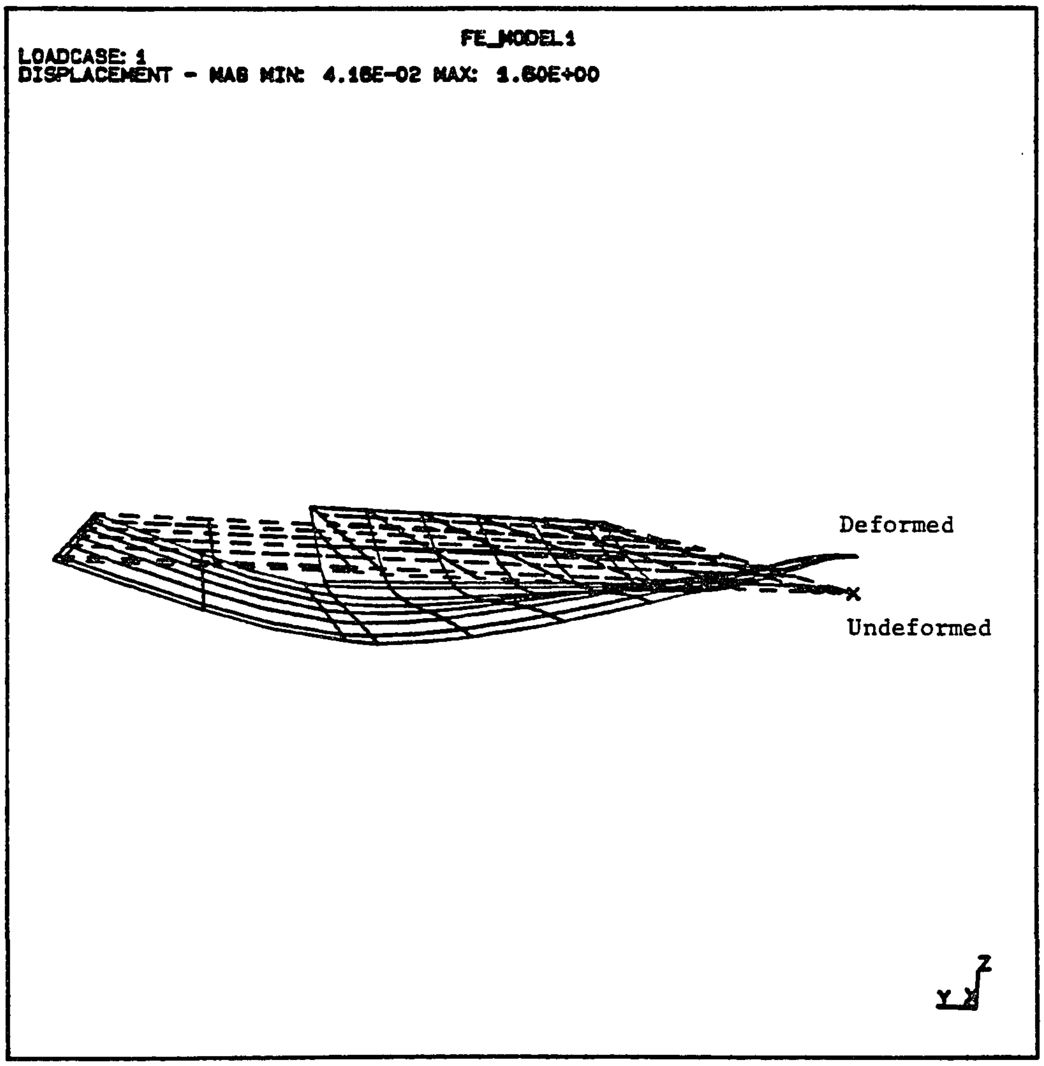


Figure 6.93: Deflection Profile, Specimen Regan Slab 4#3

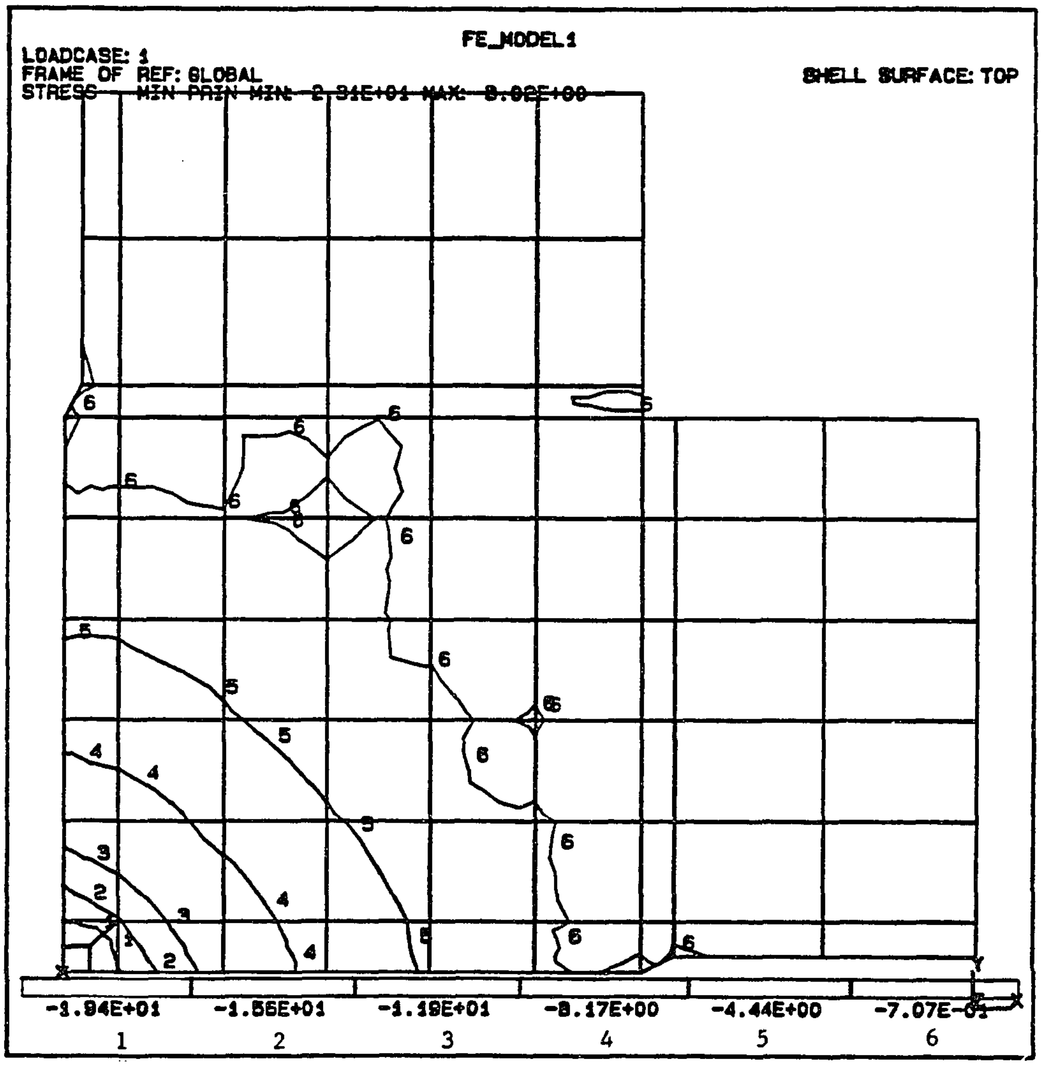


Figure 6.94: Compressive Stress Distribution, Specimen Regan Slab 4#3, Bottom Surface

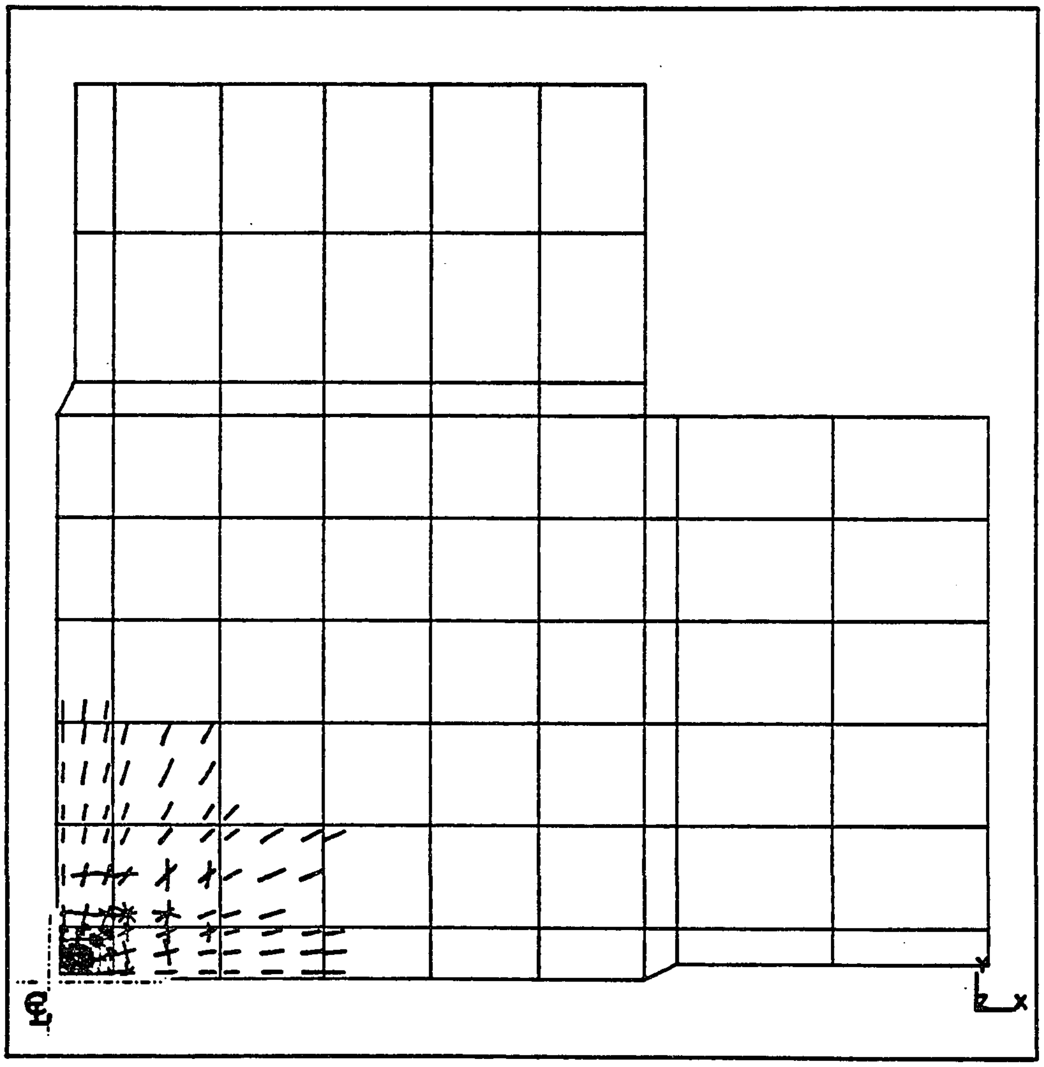


Figure 6.95: Cracking Pattern, Specimen Regan Slab 4#3, Top Surface.

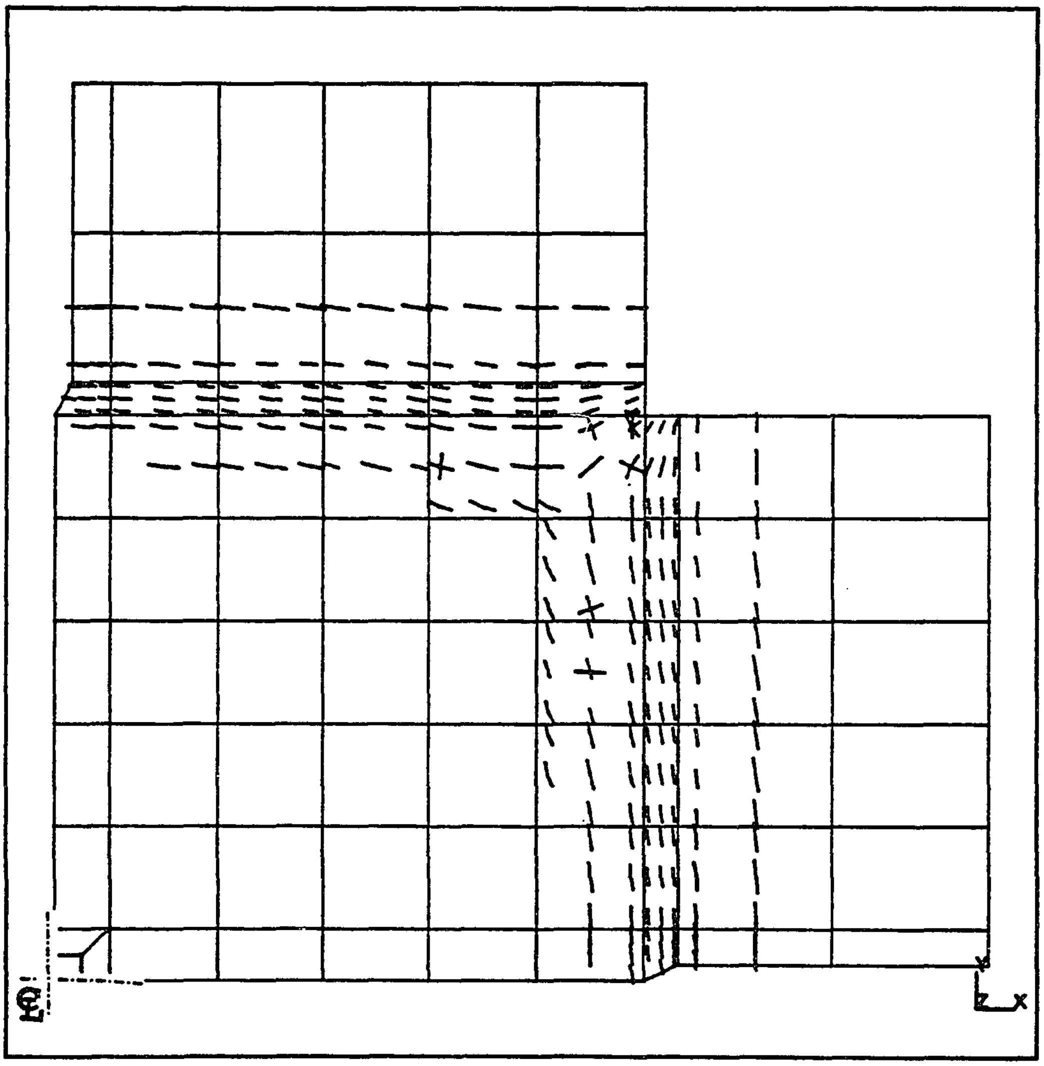


Figure 6.96: Cracking Pattern, Specimen Regan Slab 4#3, Bottom Surface.

Chapter 7

Summary and Conclusions

7.1 Summary

A three dimensional nonlinear inelastic finite element model for the analysis of reinforced concrete structures has been developed in this study. The treatment of concrete prior to cracking is simulated using a nonlinear elastic secant stiffness formulation proposed by Ottosen (1979). A strength envelope proposed by Ottosen (1977) has been employed in conjunction with this formulation to detect failure in concrete. The stress-strain response of concrete has been ratified in the present study by comparing the analytical results with available test data on concrete response to multiaxial states of stress. Steel has been simulated by employing an elasto-plastic formulation with possible strain-hardening, having uniaxial stiffness properties.

Cracking in Concrete has been simulated using a non-orthogonal multiple cracking formulation proposed by deBorst and Nauta (1985). The plain concrete stress-strain response after cracking has been simulated using a bilinear curve in the crack normal direction, similar to that employed by Rots et al. (1985). To include the tension-stiffening effects in cracked reinforced concrete, a stress strain relationship in each reinforcing direction (considered as part of concrete), has been developed in the present study. For the simulation of aggregate interlock response in reinforced concrete, a variable crack interface shear stress-strain response which simulates the effects of shear-friction has been proposed in this study. The reduction in the stiffness of the cracked concrete compressive strut, observed in tests by Vecchio and Collins (1982), has been considered in this study. Two formulations to simulate these effects have been employed in this study: one based on strains proposed by Vecchio and Collins (1982,1986) and a second one based on stresses advanced in this study.

To simulate the variations in material properties through the cross-section of an RC component, two cross-sectional layering schemes have been employed. One formulation is based on implicit procedures, similar to that by Figueras and Owen (1983), while the other is based on explicit procedures proposed by Barzegar (1989).

The capabilities of the analytical model developed in this study have been examined by analyzing the response of RC panels, walls, and slab specimen under the action of various loads.

7.2 Conclusions

The tension-stiffening formulation including its coupling effects proposed in this study is able to represent the post-cracking tensile stress-strain response of reinforced concrete test specimen adequately. The panel elements analyzed in the present study indicate that this formulation is invarient with respect to transformations (panels S1, S3, S6, - Dyngland, 1990) and size effects (bars Shima #3, Shima #5, - Shima 1987; panels PB13, - Bhide and Collins 1987, Rizkalla #2, - Rizkalla and Hwang 1984). The representation of tension-stiffening effects in each steel direction, instead of in the crack normal direction, ensures that tensile stresses are transferred through the concrete even after the reinforcement in some direction(s) have yielded (PV19 - Vecchio and Collins 1982). The present formulation is simple and depends only on uniaxial properties of concrete and steel.

The compressive stress-strain response of the cracked concrete strut is well represented by the stress-based strength and stiffness reduction procedure proposed in this study. The strain-based procedure (Vecchio and Collins 1982) tends to overestimate the reduction of strength of the strut (PV19). In addition, the strain-based procedure cannot detect the presence of bond slip (PV27, PV29 - Vecchio and Collins 1982).

The representation of the crack interface shear stress-strain response, using a variable shear stiffness procedure, proposed in this study is adequate for capturing the response of RC test specimen. The suitability of this model is demonstrated by simulating the response of uniaxially reinforced specimen (PB18 to PB22 - Bhide and Collins 1987), where the role of the crack interface shear stiffness in capturing the dominant response behavior is significant.

The layering procedures implemented in this study, both implicit and explicit, have been adequate in tracing the nonlinear response behavior through the depth of reinforced concrete specimen. The distribution of stresses through the cross-section is well represented by the implicit layering procedure, even for a moderately thick specimen, and is suitable for inclusion in material model computations. The explicit layering procedure introduces a flexibility through the cross-section due to additional degrees of freedom. This may be important for very thick cross-sections. However, the ultimate loads of the reinforced concrete specimen analyzed in this study do not appear to be altered significantly due to this shear flexibility (Delft Beam - Walraven 1978), indicating that the implicit layering procedure is suitable for nonlinear analysis to predict the ultimate loads. For thin specimen the implicit layering procedure is able to predict the displacements and strains reasonably.

The investigations in the present study indicate that the analysis and design of reinforced concrete shearwalls, which are based on uniaxial properties of concrete and steel, are inadequate in estimating the ultimate loads. It appears that the multiaxial states of stress present in the compression region of the wall need to be considered in the design of shearwalls.

The response of RC flexural and torsional specimen in the analyses clearly highlights the role of the reduced compressive strength of cracked concrete in altering the ultimate load, the load-deformational response and the mode of failure in these specimen (ML9 - Marti et al. 1987; slab S1 - Duddeck et al 1978; Regan Slab 12 and 43 - Regan 1986). Analysis procedures such as the yield line and the lower bound theory do not account for this nonlinearity and are thus unable to estimate the ultimate load adequately.

The influence of lateral restraint on the response of slab elements, in particular their "punching capacity", is well represented by the analysis in the present study up to a point. The code provisions do not account for the effect of boundary restraints in estimating the ultimate capacity of connection elements.

More investigations are needed to evaluate all the parameters influencing the load transferring capacity of the connection regions.

7.3 Recomendations for Future Work

Based on the present study, some of the areas where some further work needs to be done are identified below:

1. Experimental studies need to be conducted to determine the influence of lateral confinement on the tensile stress-strain response of RC test specimen. This would provide data to simulate 'tension-stiffening' at various levels of lateral confining stresses.

2. Develop models to consider the influence of shear dilatancy on the stress-strain response of reinforcement and tension-stiffening response. The present study considers the influence of the confining pressure due steel and tension-stiffening 'bond' on the crack interface shear response (shear-friction).

3. Develop numerical procedures that enable the use of larger load increment sizes, for use in secant stiffness models. This may involve a procedure to prevent artificial softening because of the iterative approach.

Bibliography

Abdel Rehman, H.H. (1982). "Computational Models for the Nonlinear Analysis of Reinforced Concrete Flexural Slab Systems." Ph.D thesis, University College of Swansea, U.K.

ACI (1983). " Building Code Requirements for Reinforced Concrete". ACI 318-83.

ACI (1989). " Building Code Requirements for Reinforced Concrete". ACI 318-89.

Al-Mahaidi, R.S.H. (1979). "Nonlinear Finite Element Analysis of Reinforced Concrete Deep Members." Ph.D thesis, Cornell University.

ASCE (1982). "State of the Art Report on Finite Element Analysis of Reinforced Concrete". Task Committee on Concrete and Masonry Structures, ASCE.

Ahmad, S., Irons, B.M., and Zienkiewicz, O.C. (1970). "Analysis of Thick and Thin Shell Structures, by Curved Finite Elements". IJNME, 2, pp. 419-451.

Ahmad, S.H. and Shah, S.P. (1986). "Orthotropic Model of Concrete for Triaxial Stresses". Journal of the Structural Division, ASCE. vol. 112, pp. 165-182

Al-Manseer and Phillips D.V. (1987). "Numerical Study of Some Post-cracking Parameters Affecting Nonlinear Solutions in RC Deep Beams". Proceedings of the Canadian Society of Civil Engineers. vol. 14(5). pp. 656-666.

Balakrishnan, S. and Murray, D.W.(1986). "Finite Element Predictions of RC Behavior". Structural Engineering Report 138, University of Alberta, Edmonton.

Balakrishnan, S. and Murray, D.W. (1988). "Prediction of R/C Panel and Deep Beam Behavior BY NLFEA". Journal of Structural Engineering, ASCE, vol. 114(10), pp. 2323-2342.

Balmer G.G. (1949). "Shearing Strength of Concrete Under High Triaxial Stress Computation of Mohr's Envelope as a Curve", Structural Research Laboratory Report No. SP-23, U.S. Department of Interior, Bureau of Reclamation, Denver, Colorado.

Barzegar, F. and Schnobrich, W.C.(1986). "Nonlinear Finite Element Analysis of Reinforced Concrete Under Short Term Monotonic Loading". Civil Engineering Studies SRS 530, University of Illinois, Urbana-Champaign.

Barzegar, F. (1988a). "Analysis of RC Membrane Elements with Anisotropic Reinforcement". Journal of Structural Engineering, ASCE, 115(3). pp. 647-665.

Barzegar, F. and Schnobrich, W.C.(1988b). "Modeling Tension Stiffening in Finite Element Analysis of RC Panels". Proceedings of the Annual Conference of the Canadian Society of Civil Engineers, Calgary, Canada. pp.277-303.

Barzegar, F. (1989). "Layering of RC Membrane and Plate Elements in Nonlinear Analysis". Journal of Structural Engineering, ASCE, 114(11), pp. 2474-2492.

Barzegar, F. and Ramaswamy, A. (1990). "A Secant Post-cracking Model for Reinforced Concrete with Particular Emphasis on Tension stiffening". Proceedings of the Second International Conference for Computer Aided Analysis of Concrete Structures, Austria, pp. 1001-1016.

Barzegar, F. (1991). Discussion on 'Elasto-Plastic Cracking Analysis of Reinforced Concrete' by Channakeshava, C. and Sundara Raja Iyengar, Journal of Structural Engineering, ASCE, 117(1), pp. 292-298 .

Bathe, K.J. (1982). "Finite Element Procedures in Engineering Analysis", Prentice Hall Publishers, NJ.

Bazant, Z.P. and Bhat, P. (1976). "Endochronic Theory of Inelasticity and Failure Analysis of Concrete Structures". Journal of Engineering Mechanics Division, ASCE, vol.102, pp.701-722.

Bazant, Z.P. and Kim, S.S. (1979). "Plastic Fracturing Theory for Concrete". Journal of Engineering Mechanics Division, ASCE, vol.106, pp.1287-1306.

Bazant, Z.P. (1979). "Critique of Orthotropic Models and Triaxial Testing of Concrete and Soils". Structural Engineering Report No. 79-10/640c, Northwestern University.

Bazant, Z.P. and Cedolin, L. (1979). " Blunt Crack Band Propogation in Finite Element Analysis". Journal of Engineering Mechanics, ASCE, 105(EM2), pp. 297-315.

Bazant, Z.P. and Tsubaki, T. (1979). "Total Strain Theory and Path Dependence of Concrete". Journal of Engineering mechanics division, ASCE, vol. 106, pp. 1151-1173.

Bazant, Z.P. and Shieh C.L. (1980). " Hysteretic Fracturing Endochronic Theory for Concrete ". Journal of Engineering Mechanics, ASCE, 106(EM6), pp. 929-950.

Bazant, Z.P. and Gambórova, P.(1980). "Rough Cracks in Reinforced Concrete". Journal of Structural Engineering, ASCE , 106(4), pp. 819-842.

Bazant, Z.P. and Cedolin, L. (1983). " Finite Element Modelling of Crack Band propogation". Journal of Structural Engineering, ASCE, 109(1), pp. 155-177.

Bazant, Z.P. and Oh, B.H. (1983). "Crack Band Theory for Fracture of Concrete". Materials and Structures, RILEM, Paris, vol.16, pp.155-177.

Bazant, Z.P.(1983) - "Comments on Orthotropic Models for Concrete and Geomaterials". Journal of the Engineering Mechanics Division, ASCE, 109(3), pp.849-863

Bazant, Z.P., Belytschko, T.B. and Chang, T.P. (1984). " Continuum theory of Strain Softening". Journal of Engineering Mechanics, ASCE, 110(12), pp. 1586-1610.

Belytschko, T.B. and Tsay, C.S. (1983). "A Stabilization Procedure for the Quadrilateral Plate Element with One Point Quadrature". IJNME, 19, pp. 405-419.

Bhide, S. and Collins, M.P.(1987). "Reinforced Concrete Elements in Shear and Tension". Department of Civil Engineering, University of Toronto, Publication 87-02.

Borderie, C., Berthaud, Y., and Pijaudir-cabot, G. (1990). "Crack Closure Effect in Continuum Damage Mechanics: Numerical Implementation". Second International Conference on Computer Aided Analysis and Design of Concrete Structures, Austria. pp. 975-986

British Standards Institution (1972). " Code of Practice for the Structural use of Concrete", London. CP: 110: Part 1:1972.

British Standards Institution (1985). " The Structural use of Concrete", London. BS 8110 :1985.

Bukozturk, O. and Chen, S. (1985). "Behavior Prediction of Confined Concrete Under Cyclic Loading". Finite Element Analysis of Reinforced Concrete, ASCE, C. Meyer ed., pp. 422-443.

CEB-FIP (1978). "Model Code for Concrete Structures", Comite' Euro- Internationale du Beton, Paris.

Cardenas, A. and Sozen, M. (1968). "Strength and Behavior of Isoparametrically and Non-Isoparametrically Reinforced Concrete Slabs Subjected to Combinations of Flexural and Torsional Moments". Civil Engineering Studies, CRS 336, University of Illinois, Urbana-Champaigne.

Chang, T.Y., Taniguchi, H. and Chen, W.F.(1987). "Nonlinear Finite Element Analysis of Reinforced Concrete Panels". Journal of Structural Engineering, ASCE,113(1), pp.122-140.

Channakeshava, C. (1987). "Elasto-plastic Cracking Analysis of Plain and Reinforced Concrete". Ph.D. Dissertation, Department of Civil engineering, Indian Institute of Science, Bangalore, India.

Chen, W.F. (1982). "Plasticity in Reinforced Concrete", McGraw-Hill Book Co., New York.

Cook, R.D. (1981). " Finite Element Applications to Engineering Problems", John Wiley and Sons Publishers, New Jersey.

Cope, R.J, Rao, P.V., Clark, L.A., and Norris, P. (1981). "Modelling of Reinforced Concrete Behavior for Finite Element Analysis of Bridge Slabs". Proceedings of IABSE, pp. 457-470.

Crisfield, M. and Wills, J. (1989). "Analysis of RC Panels Using Different Concrete Models". Journal of Engineering Mechanics, ASCE vol. 115(3), pp. 578-597.

Darwin, D. and Pecknold, D. (1974). "Inelastic Model for Cyclic Biaxial Loading for Reinforced Concrete". Civil Engineering Studies, SRS 409, University of Illinois, Urbana.

Darwin, D.(1986). "Concrete Crack Propagation-study of Model Parameters". Finite Element Analysis of Reinforced Concrete Structures, C.Meyer ed., ASCE, pp.184-200.

de Borst, R. and Nauta, P. (1985). "Non-Orthogonal Cracks in a Smeared Finite Element Model". Engineering Computations, 2(3), pp.35-46.

de Borst, R. (1990). "Simulation of Localization using Cosserat Theory". Second International Conference on Computer Aided Analysis and Design of Concrete Structures, Austria, pp. 931-944.

Dodds, R.H., Darwin, D., Smith, J. and Leibengood, L. (1982). "Grid Size Effects with Smeared Cracking in Finite Element Analysis of Reinforced Concrete". University of Kansas, 118 pages.

Dougill, J.W. (1976). "On Stable Progressively Fracturing Solids." ZAMP, Zeitschrift für Angewandte Mathematik und Physik, pp. 423-437.

Duddeck, H., Greibenow, G. and Schaper, G. (1978). "Material and Time Dependent Nonlinear Behavior of Cracked Reinforced Concrete Slabs", in Nonlinear Behavior of Reinforced Concrete Spatial Structures, IASS symposium, Darmstadt,G. Mehlhorn, H. Ruhle, W. Zerna ed., pp.101-113.

Dyngland, T. (1989). "Behavior of Reinforced Concrete Panels". Ph.D. dissertation, Department of Civil engineering, Norges Techniske Hogskole, Trondheim.

Eberhardsteiner, J., Meschke, G. and Mang, H.A. (1987). "Comparison of Constitutive Models for Triaxially Loaded Concrete". Proceedings of IABSE Colloquium on Computational Mechanics of Concrete Structures: Advances and Applications, pp. 197-208.

Elwi, A. and Hrudy, T. (1989). " Finite Element Model for Curved Embedded Reinforcement". ASCE, Journal of Engineering Mechanics, 115(4), pp. 740-754.

Fardis, M.N., Alibe, B. and Tassoulas, J.L. (1983). "Monotonic and Cyclic Constitutive Law for Concrete". Journal of Engineering Mechanics, ASCE, vol. 109(2), pp.516-529.

Figueras, J.A. and Owen, D.R.J. (1983). "Anisotropic Elasto-plastic Finite Element Analysis of Thick and Thin Plates and Shells". IJNME, vol.19, pp. 541-566.

Flogel, H. and Mang, H.A. (1982). "Tension Stiffening Concept Based on Bond Slip". Journal of Structural Engineering, ASCE, 108(12), pp. 2681-2701.

Foroozesh, M. (1989). "Total Lagrangian Finite Element Formulation for Metal Forming with Applications to Metal Extrusion". Ph.D. Dissertation, Department of Civil Engineering, Louisiana State University.

Frantezkakis, C. (1990). "Contribution a la Modelisation des Structures en Beton Arme par la Methode des Elements Finis". These de lENPS, Paris.

Gambarova, P.G. and Karakoc, C. (1985). "A new Approach to the Analysis of the Confinement in Regularly Cracked Concrete Elements". IABSE colloquium, H5/7, pp. 251-261, Delft.

Gerstle, K.H.(1981). "Simple Formulation for Triaxial Concrete Behavior ". ACI Journal, October, pp. 382-389.

Gilbert, R.I. and Warner, R.F.(1978). "Tension-Stiffening in Reinforced Concrete Slabs". Journal of the Structural Division, ASCE, 104(ST12), pp. 1885-1900.

Glemberg and Samualsson (1983). "A Constitutive Model for the Analysis of Reinforced Concrete Structures". Nordic Concrete Research, pp.1-40.

Gupta, A.K. and Akber H. (1984). "Cracking in Reinforced Concrete Analysis". Journal of Structural Engineering, ASCE 110(5), pp. 1735-1746.

Han, D.J. and Chen. W.F. (1985) . "Constitutive Modeling in Analysis of Concrete Structures". Report no. CE-84-19 Purdue university.

Hillerborg, A., Modeer, M. and Peterson, P.E. (1976). "Analysis of Crack Formation and Growth by means of Fracture Mechanics and Finite Elements." Cement and Concrete Research, 6(6), pp.773-782.

Hiroshi Shima, Lie-Liung, C. and Okamura, H.(1987). "Micro and Macro Models for Bond in Reinforced Concrete". Journal of the Faculty of Engineering, University of Tokyo. 39(2), pp. 133-194.

Hsieh, S., Ting, E.C., and Chen, W.F. (1979). "An Elastic Fracture Model for Concrete". Proceedings of 3rd Engineering Mechanics Division Specialty Conference, ASCE. Austin, TX. pp.437-440.

Hsieh, S., Ting, E.C., and Chen, W.F. (1988). "Application of Plastic Fracture Model to Concrete Structures". Computers and Structures, vol. 28(3), pp. 373-393.

Hu, H.T., and Schnobrich, W.C.(1988). "Nonlinear Analysis of Plane Stress State Reinforced Concrete Under Short Term Monotonic Loading". Civil Engineering Studies SRS 539, University of Illinois, Urbana-Champaign.

Ishikawa, M., Yashikawa, H. and Tanabe, T. (1985). "The Nonlinear Constitutive Model in Terms of Damage Tensor and its Application to Finite Element Method". Finite Element Analysis of Reinforced Concrete Structures. ASCE, C. Meyer ed., pp. 93-104.

Kauser, M. and Mehlhorn, G. (1987). "Finite Element Models for Bond Problems". Journal of Structural Engineering, ASCE 113(10), p.2160-2173.

Kolleger, J. and Mehlhorn, G. (1987). "Material Model for Cracked Concrete". IABSE colloquium on Computational Mechanics of Concrete Structures: Advances and Application, pp. 63-74.

Kolleger, J. and Mehlhorn, G. (1988). "Experimentalle und Analytische Untersuchungen zur Aufstellung Eines Materialmodells für Greisene Stahlbetonscheiben". University of Kassel, FRG.

Kolleger, J. (1988). "Ein Materialmodell für die Berechnung von Stahlbetonflachentragwerken". Ph.D. dissertation, University of Kassel, FRG.

Kotsovos, M.D. and Newman, J.B. (1978). "Generalised Stress-Strain Relations for Concrete". Journal of Engineering Mechanics, ASCE, vol. 104(4), pp. 213-223.

Kotsovos, M.D. (1987). "Shear Failure of Reinforced Concrete Beams". Engineering Structures. vol. 9(1), pp. 32-38.

Kotsovos, M.D. (1988). " Compressive Force Path Concept: Basis for Reinforced Concrete Ultimate Limit State Design". ACI Structural Journal, 85(S8) , pp. 68-75.

Kotsovos, M.D. (1987). "Consideration of Triaxial Stress Conditions in Design: A Necessity". ACI Structural Journal, ACI(S29), pp. 266-273.

Kupfer, H. and Gerstle, K. (1969). "Behavior of Concrete Under Biaxial Stresses". Journal of Structural Engineering, ASCE, 99(4), pp.853-866.

Kupfer, H., Hilsdorf, H.K., and Rusch, H. (1973). "Behavior of Concrete under Biaxial Stresses". ACI Journal, 66(8), pp. 656-666.

Leifas, I.D., Kotsovos, M.D. and Ambraseys, N.N. (1990a). "RC Structural Walls, Strength, Deformation Characteristics and Failure Mechanisms". ACI structural journal 87(S3) pp. 23-31.

Leifas, I.D., and Kotsovos, M.D. (1990b). " NLFE Analysis of RC Structural Walls and Design Implications". Journal of Structural Engineering, ASCE, 116(1), pp. 146-164.

Li, Baolu, Maekawa, K. and Okamura, H(1989). "Contact Density Model for Stress Transfer Across Cracks in Concrete". Journal of Faculty of Engineering, University of Tokyo. XL(1), pp. 9-52.

Linse D. and Aschl H. (1976). "Verscheule Zum Verlatten Von Beton Unter Mehrachsiger Beanspruching". Technische Universitat Munchen Lehr Stuhl Fur Massivbau, Munchen.

Liu,T.C.Y, Neilson, A.H. and Slate, F.O. (1972). "Biaxial Stress- Strain Relations for Concrete". Journal of Structural Engineering, ASCE, 98(5), pp. 1025-1034

Maestrini, S.R. and Gupta, A.K.(1987). "Membrane Behavior of Reinforced Concrete Shell Elements Including Tension-Stiffening". Reinforced Concrete Shell Research Report, North Carolina State University, Raleigh, NC.

Maier, J. and Thurliman, B. (1985)."Tests to Failure of Shearwalls". Institute of Structural Engineering, ETH Zurich.

Marti, P, Leesti, P. and Khalifa, W. (1987a). "Torsion Tests on Reinforced Concrete Slab Elements". Journal of Structural Engineering, ASCE, 113(5), pp. 994-1010.

Marti, P. and Kong, K. (1987b). "Response of Reinforced Concrete Slab Elements to Torsion". Journal of Structural Engineering, ASCE, 113(5), pp. 976-993.

Milford, R.V. and Schnobrich, W.C. (1984) . "Nonlinear Behavior of Reinforced Concrete Cooling Towers". SRS 514, Department of Civil Engineering, University of Illinois, Urbana.

Millard, S.G. and Johnson, R.P.(1984). "Shear Transfer Across Cracks in Reinforced Concrete due to Aggregate Interlock and Dowel Action". Magazine of Concrete Research, 36(126), pp. 9-21

Millard, S.G. and Johnson, R.P.(1985). "Shear Transfer in Cracked Reinforced Concrete". Magazine of Concrete Research, 37(130), pp. 3-15.

Ngo, D. and Scordelis, A. (1967). "Finite Element Analysis of RC Beams". ACI Journal. 64(3), pp. 152-163.

Noguchi, H. (1985). "Analytical Models for Cyclic Loading of RC Members". Finite Element Analysis of Reinforced Concrete, C. Meyer ed., ASCE., pp. 486-506.

Ola, D. and Ottosen, N.S. (1990). "Smeared Crack Analysis Using Generalized Fictitious Crack Model". Journal of Structural Engineering, ASCE, 116(1), pp. 55-76.

Oliver, J. (1989). "A Consistent Characteristic Length for Smeared Crack Analysis". IJNME, 28, pp. 461-474.

Oliver, J., Cervera, M., Oller, S. and Lubliner, J. (1990). "Isotropic Damage and Smeared Crack Analysis". Second International Conference on Computer Aided Analysis and Design of Concrete Structures. Austria. pp. 945-958.

Ottosen, N.S. (1977). "A Failure Criterion for Concrete". Journal of engineering mechanics, ASCE, vol.103(4), pp. 525-535.

Ottosen, N.S. (1979). "Constitutive Model for Short Time Loading of Concrete". Journal of Engineering Mechanics, ASCE, 103(4), pp. 527-535.

Ottosen, N.S. (1981). "Constitutive Modelling of Concrete versus Recent Experimental Data." Risϕ National Laboratory, Denmark, Report No. 72.

Ottosen, N.S. (1982). "Further Documentation of a Constitutive Model for Concrete". Risϕ National Laboratory, Denmark, Report No. 112.

Parisch, H. (1979). "A critical Survey of the Nine Node Degenerate Shell Element with Special Emphasis on Thin Shell and Reduced Integration." Computer Methods in Applied Mechanics and Engineering, 20, pp. 323-350.

Petersson, P.E. (1980). "Fracture Energy of Concrete: Method of Determination". Cement and Concrete Research, 10, pp. 79-89.

Pruijssers, A.F. (1988). "Aggregate Interlock and Dowel Action Under Monotonic and Cyclic Loading". Ph.D. Dissertation, Department of Civil Engineering, Delft University, Netherlands.

Ramtani, S., Berthaud, Y. and Mazars, J. (1989). "A Model for Describing Both the Anisotropic and Uniaxial Distributed Damage in Concrete". SMIRT Conference, Paris.

Rashid, Y.R. (1968). "Ultimate Strength Analysis of Prestressed Concrete Pressure Vessels". Nuclear engineering and Design, 7(4), pp. 334-344.

Razaqpur, A.G.(1988). "Nonlinear Analysis of Shear Transfer in Concrete". Proceedings of the Annual Conference of the Canadian Society of Civil Engineers, pp. 237-254.

Regan, P.E. (1986). "Symmetric Punching of Reinforced Concrete Slabs". Magazine of Concrete Research, Vol. 38(136), pp. 115-128.

Richart, F.E., Brandtzaegh, A., and Brown, R.L. (1928). "A Study of the Failure of Concrete under Combined Compressive Stresses", Bulletin 185, Engineering Experiment Station, University of Illinois, Urbana.

Rots, J.G. (1985). "Bond Slip Simulations Using Smeared Cracks and or Interface Elements". Report, Department of Civil Engineering, Delft University, Netherlands.

Rots, J.G., Nauta, P., Kusters, G.M. and Blaauwendraad, J. (1985). "Smeared Crack Approach and Fracture Localization in Concrete". HERON Report, vol. 30(1), 49 pages.

Rots, J.G.(1988). "Computational Modelling of Cracked Concrete". Ph.D Dissertation, Department of Civil Engineering, DELFT University, Netherlands.

Rizkalla, S.H. and Hwang, L.S.(1984). "Crack Prediction for Members in Uniaxial Tension". ACI Journal, 81(44), pp. 572-579.

Schickert G. and Winkler, H. (1977). " Results of Tests Concerning Strength and Strain of Concrete Subjected to Multiaxial Compressive Stresses". Deutscher Ausschuss für Stahlbeton Heft. 277, Berling.

Shirai, N. and Noguchi, H. (1989). "Compressive Deterioration of Cracked Concrete". Annual Structures Congress, San Francisco, ASCE, pp.1-10

Soroushian, P., Obaseki, K. and Choi, K. (1988). "Analysis of Aggregate Interlock Behavior at Cracks in Reinforced Concrete". Magazine of Concrete Research, vol. 40(142), pp. 43-49.

Sotomura, K. and Yoshizaki, S. (1981). "Experimental Study of a Shearwall with Numerous Small Openings". 6th SMIRT conference, J4/13, Paris.

Stankowski, T. Gerstle, K.H. (1985). "Simple Formulation of Concrete Behavior under Multiaxial Load Histories". ACI Journal, 82(2), pp. 213-221.

Tassios, T.P. and Vintzelou, E.N.(1987). "Concrete-to-Concrete Friction". Journal of Structural Engineering, ASCE, 113(4), pp. 832-849.

Torrent, R.J., Dvorkin, E.N. and Alvaredo, A.M. (1987). "A Model for Work-Hardening Plasticity and Failure of Concrete Under Multiaxial Stresses". Cement and Concrete Research, vol.(17), pp. 939-950.

Torrenti, J.M. and Djebri, B.D. (1990). "Constitutive Laws for Concrete: an Attempt of Comparison". Second International Conference on Computer Aided Analysis and Design of Concrete Structures, Austria. pp. 871-882.

Ueda, M. and Kawai, T. (1985). "Discrete Limit Analysis of RC Shearwalls". Finite Element Analysis of Reinforced Concrete Structures, ASCE, pp. 277-287.

Van Der Vorm, P.J (1988). "A Numerical Study of Reinforced Concrete Panels". M.S. thesis, Delft University, Netherlands.

Vecchio, F.J. and Collins, M.P.(1982). "The Response of Reinforced Concrete Elements to In-plane Shear and Normal Stresses". Department of Civil Engineering, University of Toronto, Publication 82-03.

Vecchio, F.J. and Collins, M.P.(1986). "The Modified Compression Field Theory for Reinforced Concrete Elements Subjected to Shear". ACI Journal, 83(22), pp. 219-231.

Vintzeleou, E.N. and Tassios, T.P.(1987). "Behavior of Dowels Under Cyclic Deformations". ACI Structural Journal, 84(S3), pp. 18-30

Walraven (1978). " The Influence of Depth on The Shear Strength of Light Weight Concrete Beams Without Shear Reinforcement, Report 5-78-4, Stevens Laboratory, Delft University, Netherlands.

Willam, K.J., Bicanic, N. and Sture, S.(1984). "Constitutive and Computational Aspects of Strain Softening and Localization in Solids". Proceedings of ASME-WAM Symposium on Constitutive Equations: Micro, Macro and Computational Aspects, New Orleans, pp.1-20.

Willam, K.J. and Warnke, E.P. (1975). "Constitutive Model for the Triaxial Behavior of Concrete". Proceeding of IABSE, vol. 19, pp. 1-30.

Wang, Q. (1988). "Finite Element Analysis and Limit Analysis of RC Panels". M.S. Thesis, Delft University, Netherlands.

Wang, Q. Van Der Vorm, P.J. and Blauwendraad, J. (1990). " Failure of RC Panels: How Accurate the Models Must Be?". Proceedings of the Second International Conference on Computer Aided Analysis of Concrete Structures, Ed. Bicanic, N and Mang, H., Austria (1990), pp. 153-163.

Yamaguchi, H. and Nomura, S. (1985). "Analysis of RC Walls by Plastic Fracturing Theories". Finite Element Analysis of Reinforced Concrete, ASCE, C. Meyer ed., pp. 204-213.

Yamaguchi, E. and Chen, W.F. (1990). "Cracking Model for Finite Element Analysis of Concrete Materials". Journal of Engineering Mechanics, 116(6), pp. 1242-1260.

Yashikawa,H., Zhishen, Wu, and Tanabe, T. (1989). "Analytical Model for Shear Slip of Cracked Concrete". Journal of Structural Engineering, ASCE, 115(4), pp. 771-786.

Yashikawa, H. and Tanabe, T. (1986). "Tension Stiffness and Constitutive Equation of a Cracked Reinforced Concrete Panel in the Plane Stress State". Trans. of Japan Concrete Institute, 8(IV-13-A), pp. 457-464.

Yashikawa, H. and Tanabe, T. (1986). "An Analytical Study of the Tension Stiffness of Reinforced Concrete Members on the Basis of Bond Slip". Trans. of Japan Concrete Institute, 8(IV-15-A), pp. 473-480.

Zienkiewicz, O.C., Taylor, R. and Too, J.M. (1971). "Reduced Integration in General Analysis of Plates and Shells." IJNME, 3, pp. 275-290.

NONLINEAR INELASTIC FINITE ELEMENT ANALYSIS OF REINFORCED CONCRETE STRUCTURES WITH EMPHASIS ON SHEAR AND TORSION
Volume II

A Dissertation

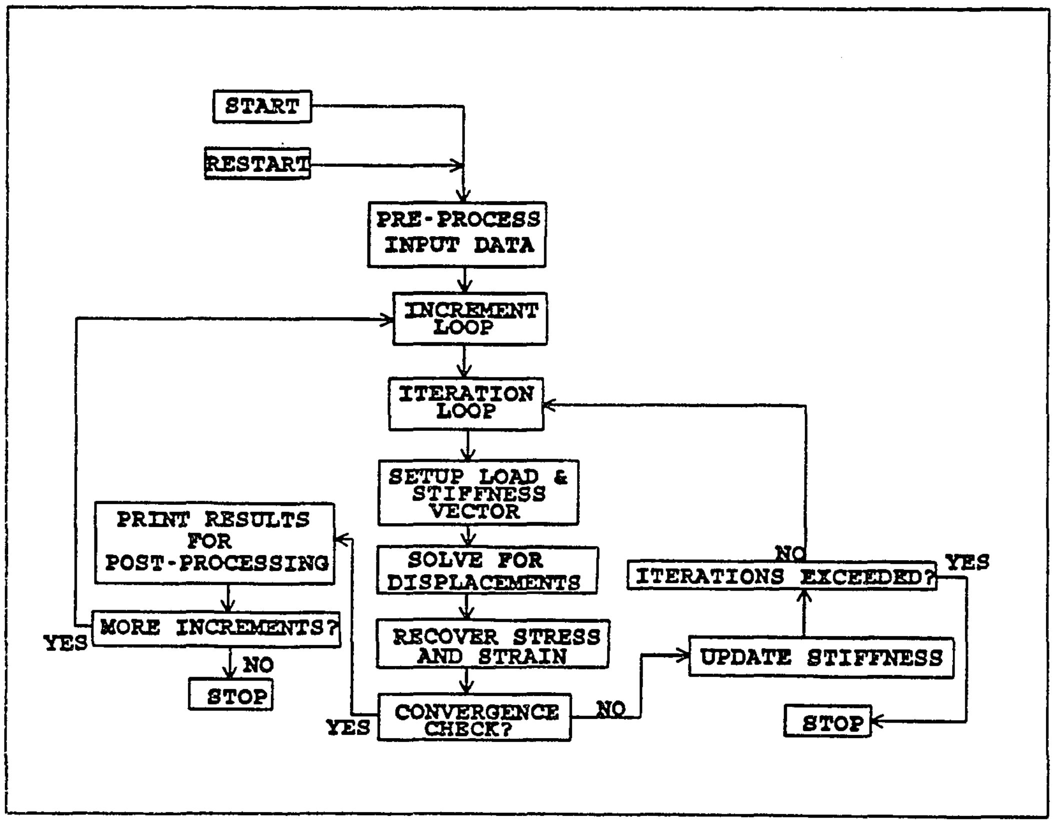
Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Civil Engineering

by
Ananth Ramaswamy
B.Tech., Indian Institute of Technology, 1985
M.S. University of California, 1986
May, 1992

Appendix A
Flow Chart



Appendix B

Sample Input Data File

```
OFP='CERAMA.DENELA' VOLSER='USER07' KEEP  ! NAMES OF OUTPUT FILES
SCFP = 'CERAMA.DENELA'              ! NAMES OF SCRATCH FILES STATUS KEEP
BEGIN   ! SETUP INPUT FILES FOR PROCESSING
DIMENSION 3  ! 3 D PROBLEM
LINEAR    ! GEOMETRIC LINEARITY
SYMMETRIC  ! SYMMETRIC SOLVER
INCREMENTS 40 FRACT 0.4000   ITERATIONS 80 ICKNTR 20 ! INCREMENTS, LOAD FRACTION
INCREMENTS 35 FRACT 0.5000   ITERATIONS 80 ICKNTR 20 ! NUMBER OF ITERATIONS
INCREMENTS 35 FRACT 0.5500   ITERATIONS 80 ICKNTR 20 ! CRACK OUTPUT EVERY 20
INCREMENTS 30 FRACT 0.6000   ITERATIONS 80 ICKNTR 20 ! STEPS.
INCREMENTS 40 FRACT 0.7000   ITERATIONS 80 ICKNTR 20
INCREMENTS 40 FRACT 0.8000   ITERATIONS 80 ICKNTR 20
INCREMENTS 35 FRACT 0.9000   ITERATIONS 80 ICKNTR 20
INCREMENTS 30 FRACT 1.0000   ITERATIONS 80 ICKNTR 20
MATERIAL 1 TYPE 4    ! MATERIAL TYPE 4 REINFORCED CONCRETE
ELEMENT  1 TO 7 TYPE 3408 MATERIAL 1  THICK 60.0  !ELEMENT TYPE AND PROPERTY
ELEMENT  8 TO 42 TYPE 3408 MATERIAL 1 THICK 78.0
CAEDS 'CERAMA.DENBEL.UNV' !CAEDS UNIVERSAL FILE PROCESSED FOR NODES AND ELEMENTS
LOADS ! LOADING TYPE AND INFORMATION
    NODE 216 FZ  -13333.3
    NODE 217 FZ  -53333.3
    NODE 218 FZ  -13333.3
END
OUTPUT 1 DISPLACEMENTS REACTIONS CONCRETE  ! OUTPUT INFO FOR REINFORCED CONCRETE
ELEM 1 TO 42 DONE NODE 1 TO 228 DONE END  ! AT SPECIFIC ELEMENTS AND NODES
RCPREP ! CALL TO MATERIAL MODEL INPUT PROCESSOR
SECANT ! SECANT MODEL
MATERIAL ! MATERIAL 1 TYPE - E.G. CONCRETE  AND INFORMATION
CONCRETE

E 28000.0 NU 0.20 FC 28.0  FT 2.50 EPSC 0.002  EPSU 0.0030 DSOFT 0.0001
BSHR 0.99 GF 0.40 NUF 0.45 BETAA 0.8 FCSIG 0.85 ATHETA 60. TSFLAG -99.0
TSTERM 0.003 TOLER 10. TSTIFG 1 CSOFTN 2
MATERIAL ! MATERIAL 2 TYPE - STEEL AND INFORMATION
STEEL
E 210000.0 NU 0.0 YEILD 440.0 AHARD 0.001 EPSBRK 0.022
ELEM 1 TO  7    ! LAYERING INFORMATION-NUMBER OF INTERNAL LAYERS AND DESCRIPTION
INTLYR 5        ! FOR THE ABOVE GROUP OF ELEMENTS E.G. 1 TO 5.
LMAT 1 LTHICK 13.45 ZI -23.275 NIPXI 3 NIPETA 3 NIPSI 2 !MATERIAL NUMBER,
ORIENT 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 ! LOCATION ZI,NO. OF GAUSS PTS.,
LMAT 1 LTHICK 15.00 ZI -9.05 NIPXI 3 NIPETA 3 NIPSI 2 !ORIENTATION OF LAYER ETC.
ORIENT 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0
LMAT 2 LTHICK  3.1 ZI 0.0 NIPXI 3 NIPETA 3 NIPSI 2
ORIENT 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0
LMAT 1 LTHICK 15.00 ZI  9.05 NIPXI 3 NIPETA 3 NIPSI 2
ORIENT 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0
LMAT 1 LTHICK 13.45 ZI 23.275 NIPXI 3 NIPETA 3 NIPSI 2
ORIENT 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0
DONE
ELEM 8 TO 42  ! REMAINING ELEMENTS HAVE ONE LAYER OF TYPE 1 -CONCRETE
INTLYR 1
LMAT 1 LTHICK 78.00 ZI 0.0 NIPXI 3 NIPETA 3 NIPSI 2
ORIENT 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0
DONE
END
CONSTRAINT    ! MULTIPOINT CONSTRAINING DONE EXTERNALLY.
```

```
STACK ELEM 36 TO 1 BY -7 DONE STOP ! COULD ALSO BE DONE FROM WITHIN CAEDS.
STACK ELEM 37 TO 2 BY -7 DONE STOP
STACK ELEM 38 TO 3 BY -7 DONE STOP
STACK ELEM 39 TO 4 BY -7 DONE STOP
STACK ELEM 40 TO 5 BY -7 DONE STOP
STACK ELEM 41 TO 6 BY -7 DONE STOP
STACK ELEM 42 TO 7 BY -7 DONE STOP
END
```

Appendix C

Sample Caeds Universal File For Input Data

```
    -1
   151
NONE
NONE
CAEDS V3R1M1:  MONITOR
21-FEB-92   09:18:13
NEVER     NEVER
CAEDS V3R1M1:  PRE/POST pROCESSINg
21-FEB-92   09:41:04
    -1
    -1
   164
         1METRIC ABS (SI)               2
  1.00000000000000000D+00  1.00000000000000000D+00  1.00000000000000000D+00
  2.73149902343750000D+02
    -1
    -1
   154
         1ONE,    FULL SCREEN
         1
  0.00000E+00  1.00000E+00  0.00000E+00  1.00000E+00
    -1
    -1
   154
         2TWO,    SIDE BY SIDE
         2
  0.00000E+00  4.98500E-01  0.00000E+00  1.00000E+00
  5.01500E-01  1.00000E+00  0.00000E+00  1.00000E+00
    -1
    -1
   154
         3THREE, LARGE LEFT
         3
  0.00000E+00  6.58500E-01  0.00000E+00  1.00000E+00
  6.61500E-01  1.00000E+00  5.02000E-01  1.00000E+00
  6.61500E-01  1.00000E+00  0.00000E+00  4.98000E-01
    -1
    -1
   154
         4FOUR,   EQUAL SIZE
         4
  0.00000E+00  4.98500E-01  5.02000E-01  1.00000E+00
  5.01500E-01  1.00000E+00  5.02000E-01  1.00000E+00
  0.00000E+00  4.98500E-01  0.00000E+00  4.98000E-01
  5.01500E-01  1.00000E+00  0.00000E+00  4.98000E-01
    -1
    -1
   160
         1OBSERVER
         1         15          0          0          0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
  0.00000E+00  1.00000E+00
    -1
    -1
```

```
 160
      2RIGHT_SHOULDER
      0        15         2         2         0
0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
0.00000E+00  3.46000E+00
  -1
  -1
 160
      3LEFT_SHOULDER
      0        15         2         2         0
0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
0.00000E+00  3.46000E+00
  -1
  -1
 436
      0BLACK
0.00000E+00  0.00000E+00  0.00000E+00
  -1
  -1
 436
      1BLUE
0.00000E+00  0.00000E+00  1.00000E+00
  -1
  -1
 436
      2GRAY_BLUE
0.00000E+00  3.30000E-01  1.00000E+00
  -1
  -1
 436
      3LIGHT_BLUE
0.00000E+00  6.60000E-01  1.00000E+00
  -1
  -1
 436
      4CYAN
0.00000E+00  1.00000E+00  1.00000E+00
  -1
  -1
 436
      5DARK_OLIVE


0.00000E+00  3.30000E-01  0.00000E+00
  -1
  -1
 436
      6DARK_GREEN
0.00000E+00  6.60000E-01  0.00000E+00
  -1
  -1
 436
      7GREEN
0.00000E+00  1.00000E+00  0.00000E+00
  -1
  -1
 436
      8YELLOW
1.00000E+00  1.00000E+00  0.00000E+00
  -1
  -1
 436
```

```
       9GOLDEN_ORANGE
 1.00000E+00  6.60000E-01  0.00000E+00
   -1
   -1
  436
      10ORANGE
 1.00000E+00  3.30000E-01  0.00000E+00
   -1
   -1
  436
      11RED
 1.00000E+00  0.00000E+00  0.00000E+00
   -1
   -1
  436
      12MAGENTA
 1.00000E+00  0.00000E+00  1.00000E+00
   -1
   -1
  436
      13LIGHT_MAGENTA
 1.00000E+00  3.30000E-01  1.00000E+00
   -1
   -1
  436
      14PINK
 1.00000E+00  6.60000E-01  1.00000E+00
   -1
   -1
  436
      15WHITE
 1.00000E+00  1.00000E+00  1.00000E+00
   -1
   -1
  800
       1
WORKING_SET1
   -1
   -1
  801
       1         0         1         8         1         1         0         0
       0         0         0         1         0
   1     1
  0.0000000000000000D+00  0.0000000000000000D+00  0.0000000000000000D+00
  2.9000000000000000D+02  0.0000000000000000D+00  0.0000000000000000D+00
       2         0         1         8         1         1         0         0
       0         0         0         1         0
   1     1
  2.9000000000000000D+02  0.0000000000000000D+00  0.0000000000000000D+00
  1.5500000000000000D+03  0.0000000000000000D+00  0.0000000000000000D+00
       3         0         1         8         1         1         0         0
       0         0         0         1         0
   1     1
  1.5500000000000000D+03  0.0000000000000000D+00  0.0000000000000000D+00
  2.0500000000000000D+03  0.0000000000000000D+00  0.0000000000000000D+00
       4         0         1         8         1         1         0         0
       0         0         0         1         0
   1     1
  2.0500000000000000D+03  0.0000000000000000D+00  0.0000000000000000D+00
  2.0500000000000000D+03  2.0000000000000000D+02  0.0000000000000000D+00
       5         0         1         8         1         1         0         0
       0         0         0         1         0
   1     1
```

```
  2.0500000000000000D+03  2.0000000000000000D+02  0.0000000000000000D+00
  1.5500000000000000D+03  2.0000000000000000D+02  0.0000000000000000D+00
         6         0         1         8         1         1         0         0
         0         0         0         1         0
     1     1
  1.5500000000000000D+03  2.0000000000000000D+02  0.0000000000000000D+00
  2.9000000000000000D+02  2.0000000000000000D+02  0.0000000000000000D+00
         7         0         1         8         1         1         0         0
         0         0         0         1         0
     1     1
  2.9000000000000000D+02  2.0000000000000000D+02  0.0000000000000000D+00
  0.0000000000000000D+00  2.0000000000000000D+02  0.0000000000000000D+00
         8         0         1         8         1         1         0         0
         0         0         0         1         0
     1     1

  0.0000000000000000D+00  2.0000000000000000D+02  0.0000000000000000D+00
  0.0000000000000000D+00  0.0000000000000000D+00  0.0000000000000000D+00
         9         0         1         8         1         1         0         0
         0         0         0         1         0
     1     1
  2.9000000000000000D+02  0.0000000000000000D+00  0.0000000000000000D+00
  2.9000000000000000D+02  2.0000000000000000D+02  0.0000000000000000D+00
        10         0         1         8         1         1         0         0
         0         0         0         1         0
     1     1
  1.5500000000000000D+03  0.0000000000000000D+00  0.0000000000000000D+00
  1.5500000000000000D+03  2.0000000000000000D+02  0.0000000000000000D+00
    -1

    -1
   802
         1         0         1         7         2
  0.0000000000000000D+00  0.0000000000000000D+00  0.0000000000000000D+00
         2         0         1         7         2
  2.9000000000000000D+02  0.0000000000000000D+00  0.0000000000000000D+00
         3         0         1         7         2
  1.5500000000000000D+03  0.0000000000000000D+00  0.0000000000000000D+00
         4         0         1         7         2
  2.0500000000000000D+03  0.0000000000000000D+00  0.0000000000000000D+00
         5         0         1         7         2
  2.0500000000000000D+03  2.0000000000000000D+02  0.0000000000000000D+00
         6         0         1         7         2
  1.5500000000000000D+03  2.0000000000000000D+02  0.0000000000000000D+00
         7         0         1         7         2
  2.9000000000000000D+02  2.0000000000000000D+02  0.0000000000000000D+00
         8         0         1         7         2
  0.0000000000000000D+00  2.0000000000000000D+02  0.0000000000000000D+00
    -1
    -1
   749
         1         1
FE_MODEL1
    -1
    -1
   731
         1        90         7
PHYSICAL PROPERTY TABLE1
 1.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 1.000000E-35 1.000000E-35
 1.000000E-35 1.000000E-35 1.000000E-35 0.000000E+00 0.000000E+00 0.000000E+00
 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00
 0.000000E+00
    -1
```

```
   -1
  747
        1         1        27
MATERIAL PROPERTY TABLE1
 2.068000E+11 2.900000E-01 7.820000E+03 1.000000E-35 3.610000E-06 1.000000E-35
 1.000000E-35 4.500000E+01 1.000000E-35 0.000000E+00 2.480000E+08 0.000000E+00
 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 1.000000E+00
 1.000000E+00 1.000000E+00 1.000000E-04 1.500000E+09 1.500000E+09 1.500000E+09
 1.500000E+09 6.800000E+07 0.000000E+00
   -1
   -1
   92
        1         2         2         4         1
        2         2         1         1         2         2
        1         1         0         0         8         0
  0.00000E+00  2.00000E+02  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  1.00000E+00  2.90000E+02  2.00000E+02  0.00000E+00  1.00000E+00
  2.90000E+02  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00
  1.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  1.00000E+00
        2         2         2         4         1
        2         2         1         1         2         2
        1         1         0         0         8         0
  2.90000E+02  2.00000E+02  0.00000E+00  1.00000E+00  2.90000E+02  0.00000E+00
  0.00000E+00  1.00000E+00  1.55000E+03  2.00000E+02  0.00000E+00  1.00000E+00
  1.55000E+03  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00
  1.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  1.00000E+00
        3         2         2         4         1
        2         2         1         1         2         2
        1         1         0         0         8         0
  1.55000E+03  2.00000E+02  0.00000E+00  1.00000E+00  1.55000E+03  0.00000E+00
  0.00000E+00  1.00000E+00  2.05000E+03  2.00000E+02  0.00000E+00  1.00000E+00
  2.05000E+03  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00
  1.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  1.00000E+00
   -1
   -1
  743
        1        -1         1  0.00000E+00
        1
        2        -1         2  0.00000E+00
        1         2
        3        -1         2  0.00000E+00
        2         3
        4        -1         1  0.00000E+00
        3
        5        -1         1  0.00000E+00
        3
        6        -1         2  0.00000E+00
        2         3
        7        -1         2  0.00000E+00
        1         2

        8        -1         1  0.00000E+00
        1
   -1
   -1
  744
        1         1         2         1         1
  1.45000E+02  0.00000E+00  0.00000E+00
        1
        2         2         3         1         1
  9.20000E+02  0.00000E+00  0.00000E+00
        2
        3         3         4         1         1
```

```
1.80000E+03  0.00000E+00  0.00000E+00
     3
     4         4         5         1         1
2.05000E+03  1.00000E+02  0.00000E+00
     3
     5         5         6         1         1
1.80000E+03  2.00000E+02  0.00000E+00
     3
     6         6         7         1         1
9.20000E+02  2.00000E+02  0.00000E+00
     2
     7         7         8         1         1
1.45000E+02  2.00000E+02  0.00000E+00
     1
     8         8         1         1         1
0.00000E+00  1.00000E+02  0.00000E+00
     1
     9         2         7         2         1
2.90000E+02  1.00000E+02  0.00000E+00
     1         2
    10         3         6         2         1
1.55000E+03  1.00000E+02  0.00000E+00
     2         3
  -1
  -1
 761
2.54000E-02
  -1
  -1
 705
         1        12         1         0         0        11         2         4
        95         1         1         7         1         1         1         0
         8         1         9         7
         1
         2        12         1         0         0        11         2         4
        95         1         1         7         1         1         2         0
         9         2        10         6
        -1
         3        12         1         0         0        11         2         4
        95         1         1         7         1         1         3         0
        10         3         4         5
        -1
  -1
  -1
  15
         1         0         0        11 0.000000E+00 2.000000E+02 0.000000E+00
         2         0         0        11 0.000000E+00 1.000000E+02 0.000000E+00
         3         0         0        11 0.000000E+00 0.000000E+00 0.000000E+00
         4         0         0        11 1.450000E+02 2.000000E+02 0.000000E+00
         5         0         0        11 1.450000E+02 0.000000E+00 0.000000E+00
         6         0         0        11 2.900000E+02 2.000000E+02 0.000000E+00
         7         0         0        11 2.900000E+02 1.000000E+02 0.000000E+00
         8         0         0        11 2.900000E+02 0.000000E+00 0.000000E+00
         9         0         0        11 4.475000E+02 2.000000E+02 0.000000E+00
        10         0         0        11 4.475000E+02 0.000000E+00 0.000000E+00
        11         0         0        11 6.050000E+02 2.000000E+02 0.000000E+00
        12         0         0        11 6.050000E+02 1.000000E+02 0.000000E+00
        13         0         0        11 6.050000E+02 0.000000E+00 0.000000E+00
        14         0         0        11 7.625000E+02 2.000000E+02 0.000000E+00
        15         0         0        11 7.625000E+02 0.000000E+00 0.000000E+00
        16         0         0        11 9.200000E+02 2.000000E+02 0.000000E+00
        17         0         0        11 9.200000E+02 1.000000E+02 0.000000E+00
        18         0         0        11 9.200000E+02 0.000000E+00 0.000000E+00
```

19	0	0	11	1.077500E+03	2.000000E+02	0.000000E+00
20	0	0	11	1.077500E+03	0.000000E+00	0.000000E+00
21	0	0	11	1.235000E+03	2.000000E+02	0.000000E+00
22	0	0	11	1.235000E+03	1.000000E+02	0.000000E+00
23	0	0	11	1.235000E+03	0.000000E+00	0.000000E+00
24	0	0	11	1.392500E+03	2.000000E+02	0.000000E+00
25	0	0	11	1.392500E+03	0.000000E+00	0.000000E+00
26	0	0	11	1.550000E+03	2.000000E+02	0.000000E+00
27	0	0	11	1.550000E+03	1.000000E+02	0.000000E+00
28	0	0	11	1.550000E+03	0.000000E+00	0.000000E+00
29	0	0	11	1.675000E+03	2.000000E+02	0.000000E+00
30	0	0	11	1.675000E+03	0.000000E+00	0.000000E+00
31	0	0	11	1.800000E+03	2.000000E+02	0.000000E+00
32	0	0	11	1.800000E+03	1.000000E+02	0.000000E+00
33	0	0	11	1.800000E+03	0.000000E+00	0.000000E+00
34	0	0	11	1.925000E+03	2.000000E+02	0.000000E+00
35	0	0	11	1.925000E+03	0.000000E+00	0.000000E+00
36	0	0	11	2.050000E+03	2.000000E+02	0.000000E+00
37	0	0	11	2.050000E+03	1.000000E+02	0.000000E+00
38	0	0	11	2.050000E+03	0.000000E+00	0.000000E+00
39	0	0	11	0.000000E+00	2.000000E+02	6.900000E+01
40	0	0	11	0.000000E+00	1.000000E+02	6.900000E+01
41	0	0	11	0.000000E+00	0.000000E+00	6.900000E+01
42	0	0	11	1.450000E+02	2.000000E+02	6.900000E+01
43	0	0	11	1.450000E+02	0.000000E+00	6.900000E+01
44	0	0	11	2.900000E+02	2.000000E+02	6.900000E+01
45	0	0	11	2.900000E+02	1.000000E+02	6.900000E+01
46	0	0	11	2.900000E+02	0.000000E+00	6.900000E+01
47	0	0	11	4.475000E+02	2.000000E+02	6.900000E+01
48	0	0	11	4.475000E+02	0.000000E+00	6.900000E+01
49	0	0	11	6.050000E+02	2.000000E+02	6.900000E+01
50	0	0	11	6.050000E+02	1.000000E+02	6.900000E+01
51	0	0	11	6.050000E+02	0.000000E+00	6.900000E+01
52	0	0	11	7.625000E+02	2.000000E+02	6.900000E+01
53	0	0	11	7.625000E+02	0.000000E+00	6.900000E+01
54	0	0	11	9.200000E+02	2.000000E+02	6.900000E+01
55	0	0	11	9.200000E+02	1.000000E+02	6.900000E+01
56	0	0	11	9.200000E+02	0.000000E+00	6.900000E+01
57	0	0	11	1.077500E+03	2.000000E+02	6.900000E+01
58	0	0	11	1.077500E+03	0.000000E+00	6.900000E+01
59	0	0	11	1.235000E+03	2.000000E+02	6.900000E+01
60	0	0	11	1.235000E+03	1.000000E+02	6.900000E+01
61	0	0	11	1.235000E+03	0.000000E+00	6.900000E+01
62	0	0	11	1.392500E+03	2.000000E+02	6.900000E+01
63	0	0	11	1.392500E+03	0.000000E+00	6.900000E+01
64	0	0	11	1.550000E+03	2.000000E+02	6.900000E+01
65	0	0	11	1.550000E+03	1.000000E+02	6.900000E+01
66	0	0	11	1.550000E+03	0.000000E+00	6.900000E+01
67	0	0	11	1.675000E+03	2.000000E+02	6.900000E+01
68	0	0	11	1.675000E+03	0.000000E+00	6.900000E+01
69	0	0	11	1.800000E+03	2.000000E+02	6.900000E+01
70	0	0	11	1.800000E+03	1.000000E+02	6.900000E+01
71	0	0	11	1.800000E+03	0.000000E+00	6.900000E+01
72	0	0	11	1.925000E+03	2.000000E+02	6.900000E+01
73	0	0	11	1.925000E+03	0.000000E+00	6.900000E+01
74	0	0	11	2.050000E+03	2.000000E+02	6.900000E+01
75	0	0	11	2.050000E+03	1.000000E+02	6.900000E+01
76	0	0	11	2.050000E+03	0.000000E+00	6.900000E+01
77	0	0	11	0.000000E+00	2.000000E+02	1.470000E+02
78	0	0	11	0.000000E+00	1.000000E+02	1.470000E+02
79	0	0	11	0.000000E+00	0.000000E+00	1.470000E+02
80	0	0	11	1.450000E+02	2.000000E+02	1.470000E+02

81	0	0	11	1.450000E+02	0.000000E+00	1.470000E+02
82	0	0	11	2.900000E+02	2.000000E+02	1.470000E+02
83	0	0	11	2.900000E+02	1.000000E+02	1.470000E+02
84	0	0	11	2.900000E+02	0.000000E+00	1.470000E+02
85	0	0	11	4.475000E+02	2.000000E+02	1.470000E+02
86	0	0	11	4.475000E+02	0.000000E+00	1.470000E+02
87	0	0	11	6.050000E+02	2.000000E+02	1.470000E+02
88	0	0	11	6.050000E+02	1.000000E+02	1.470000E+02
89	0	0	11	6.050000E+02	0.000000E+00	1.470000E+02
90	0	0	11	7.625000E+02	2.000000E+02	1.470000E+02
91	0	0	11	7.625000E+02	0.000000E+00	1.470000E+02
92	0	0	11	9.200000E+02	2.000000E+02	1.470000E+02
93	0	0	11	9.200000E+02	1.000000E+02	1.470000E+02
94	0	0	11	9.200000E+02	0.000000E+00	1.470000E+02
95	0	0	11	1.077500E+03	2.000000E+02	1.470000E+02
96	0	0	11	1.077500E+03	0.000000E+00	1.470000E+02
97	0	0	11	1.235000E+03	2.000000E+02	1.470000E+02
98	0	0	11	1.235000E+03	1.000000E+02	1.470000E+02
99	0	0	11	1.235000E+03	0.000000E+00	1.470000E+02
100	0	0	11	1.392500E+03	2.000000E+02	1.470000E+02
101	0	0	11	1.392500E+03	0.000000E+00	1.470000E+02
102	0	0	11	1.550000E+03	2.000000E+02	1.470000E+02
103	0	0	11	1.550000E+03	1.000000E+02	1.470000E+02
104	0	0	11	1.550000E+03	0.000000E+00	1.470000E+02
105	0	0	11	1.675000E+03	2.000000E+02	1.470000E+02
106	0	0	11	1.675000E+03	0.000000E+00	1.470000E+02
107	0	0	11	1.800000E+03	2.000000E+02	1.470000E+02
108	0	0	11	1.800000E+03	1.000000E+02	1.470000E+02
109	0	0	11	1.800000E+03	0.000000E+00	1.470000E+02
110	0	0	11	1.925000E+03	2.000000E+02	1.470000E+02
111	0	0	11	1.925000E+03	0.000000E+00	1.470000E+02
112	0	0	11	2.050000E+03	2.000000E+02	1.470000E+02
113	0	0	11	2.050000E+03	1.000000E+02	1.470000E+02
114	0	0	11	2.050000E+03	0.000000E+00	1.470000E+02
115	0	0	11	0.000000E+00	2.000000E+02	2.250000E+02
116	0	0	11	0.000000E+00	1.000000E+02	2.250000E+02
117	0	0	11	0.000000E+00	0.000000E+00	2.250000E+02
118	0	0	11	1.450000E+02	2.000000E+02	2.250000E+02
119	0	0	11	1.450000E+02	0.000000E+00	2.250000E+02
120	0	0	11	2.900000E+02	2.000000E+02	2.250000E+02
121	0	0	11	2.900000E+02	1.000000E+02	2.250000E+02
122	0	0	11	2.900000E+02	0.000000E+00	2.250000E+02
123	0	0	11	4.475000E+02	2.000000E+02	2.250000E+02
124	0	0	11	4.475000E+02	0.000000E+00	2.250000E+02
125	0	0	11	6.050000E+02	2.000000E+02	2.250000E+02
126	0	0	11	6.050000E+02	1.000000E+02	2.250000E+02
127	0	0	11	6.050000E+02	0.000000E+00	2.250000E+02
128	0	0	11	7.625000E+02	2.000000E+02	2.250000E+02
129	0	0	11	7.625000E+02	0.000000E+00	2.250000E+02
130	0	0	11	9.200000E+02	2.000000E+02	2.250000E+02
131	0	0	11	9.200000E+02	1.000000E+02	2.250000E+02
132	0	0	11	9.200000E+02	0.000000E+00	2.250000E+02
133	0	0	11	1.077500E+03	2.000000E+02	2.250000E+02
134	0	0	11	1.077500E+03	0.000000E+00	2.250000E+02
135	0	0	11	1.235000E+03	2.000000E+02	2.250000E+02
136	0	0	11	1.235000E+03	1.000000E+02	2.250000E+02
137	0	0	11	1.235000E+03	0.000000E+00	2.250000E+02
138	0	0	11	1.392500E+03	2.000000E+02	2.250000E+02
139	0	0	11	1.392500E+03	0.000000E+00	2.250000E+02
140	0	0	11	1.550000E+03	2.000000E+02	2.250000E+02
141	0	0	11	1.550000E+03	1.000000E+02	2.250000E+02
142	0	0	11	1.550000E+03	0.000000E+00	2.250000E+02

143	0	0	11	1.675000E+03	2.000000E+02	2.250000E+02
144	0	0	11	1.675000E+03	0.000000E+00	2.250000E+02
145	0	0	11	1.800000E+03	2.000000E+02	2.250000E+02
146	0	0	11	1.800000E+03	1.000000E+02	2.250000E+02
147	0	0	11	1.800000E+03	0.000000E+00	2.250000E+02
148	0	0	11	1.925000E+03	2.000000E+02	2.250000E+02
149	0	0	11	1.925000E+03	0.000000E+00	2.250000E+02
150	0	0	11	2.050000E+03	2.000000E+02	2.250000E+02
151	0	0	11	2.050000E+03	1.000000E+02	2.250000E+02
152	0	0	11	2.050000E+03	0.000000E+00	2.250000E+02
153	0	0	11	0.000000E+00	2.000000E+02	3.030000E+02
154	0	0	11	0.000000E+00	1.000000E+02	3.030000E+02
155	0	0	11	0.000000E+00	0.000000E+00	3.030000E+02
156	0	0	11	1.450000E+02	2.000000E+02	3.030000E+02
157	0	0	11	1.450000E+02	0.000000E+00	3.030000E+02
158	0	0	11	2.900000E+02	2.000000E+02	3.030000E+02
159	0	0	11	2.900000E+02	1.000000E+02	3.030000E+02
160	0	0	11	2.900000E+02	0.000000E+00	3.030000E+02
161	0	0	11	4.475000E+02	2.000000E+02	3.030000E+02
162	0	0	11	4.475000E+02	0.000000E+00	3.030000E+02
163	0	0	11	6.050000E+02	2.000000E+02	3.030000E+02
164	0	0	11	6.050000E+02	1.000000E+02	3.030000E+02
165	0	0	11	6.050000E+02	0.000000E+00	3.030000E+02
166	0	0	11	7.625000E+02	2.000000E+02	3.030000E+02
167	0	0	11	7.625000E+02	0.000000E+00	3.030000E+02
168	0	0	11	9.200000E+02	2.000000E+02	3.030000E+02
169	0	0	11	9.200000E+02	1.000000E+02	3.030000E+02
170	0	0	11	9.200000E+02	0.000000E+00	3.030000E+02
171	0	0	11	1.077500E+03	2.000000E+02	3.030000E+02
172	0	0	11	1.077500E+03	0.000000E+00	3.030000E+02
173	0	0	11	1.235000E+03	2.000000E+02	3.030000E+02
174	0	0	11	1.235000E+03	1.000000E+02	3.030000E+02
175	0	0	11	1.235000E+03	0.000000E+00	3.030000E+02
176	0	0	11	1.392500E+03	2.000000E+02	3.030000E+02
177	0	0	11	1.392500E+03	0.000000E+00	3.030000E+02
178	0	0	11	1.550000E+03	2.000000E+02	3.030000E+02
179	0	0	11	1.550000E+03	1.000000E+02	3.030000E+02
180	0	0	11	1.550000E+03	0.000000E+00	3.030000E+02
181	0	0	11	1.675000E+03	2.000000E+02	3.030000E+02
182	0	0	11	1.675000E+03	0.000000E+00	3.030000E+02
183	0	0	11	1.800000E+03	2.000000E+02	3.030000E+02
184	0	0	11	1.800000E+03	1.000000E+02	3.030000E+02
185	0	0	11	1.800000E+03	0.000000E+00	3.030000E+02
186	0	0	11	1.925000E+03	2.000000E+02	3.030000E+02
187	0	0	11	1.925000E+03	0.000000E+00	3.030000E+02
188	0	0	11	2.050000E+03	2.000000E+02	3.030000E+02
189	0	0	11	2.050000E+03	1.000000E+02	3.030000E+02
190	0	0	11	2.050000E+03	0.000000E+00	3.030000E+02
191	0	0	11	0.000000E+00	2.000000E+02	3.810000E+02
192	0	0	11	0.000000E+00	1.000000E+02	3.810000E+02
193	0	0	11	0.000000E+00	0.000000E+00	3.810000E+02
194	0	0	11	1.450000E+02	2.000000E+02	3.810000E+02
195	0	0	11	1.450000E+02	0.000000E+00	3.810000E+02
196	0	0	11	2.900000E+02	2.000000E+02	3.810000E+02
197	0	0	11	2.900000E+02	1.000000E+02	3.810000E+02
198	0	0	11	2.900000E+02	0.000000E+00	3.810000E+02
199	0	0	11	4.475000E+02	2.000000E+02	3.810000E+02
200	0	0	11	4.475000E+02	0.000000E+00	3.810000E+02
201	0	0	11	6.050000E+02	2.000000E+02	3.810000E+02
202	0	0	11	6.050000E+02	1.000000E+02	3.810000E+02
203	0	0	11	6.050000E+02	0.000000E+00	3.810000E+02
204	0	0	11	7.625000E+02	2.000000E+02	3.810000E+02
205	0	0	11	7.625000E+02	0.000000E+00	3.810000E+02

```
     206         0         0        11 9.200000E+02 2.000000E+02 3.810000E+02
     207         0         0        11 9.200000E+02 1.000000E+02 3.810000E+02
     208         0         0        11 9.200000E+02 0.000000E+00 3.810000E+02
     209         0         0        11 1.077500E+03 2.000000E+02 3.810000E+02
     210         0         0        11 1.077500E+03 0.000000E+00 3.810000E+02
     211         0         0        11 1.235000E+03 2.000000E+02 3.810000E+02
     212         0         0        11 1.235000E+03 1.000000E+02 3.810000E+02
     213         0         0        11 1.235000E+03 0.000000E+00 3.810000E+02
     214         0         0        11 1.392500E+03 2.000000E+02 3.810000E+02
     215         0         0        11 1.392500E+03 0.000000E+00 3.810000E+02
     216         0         0        11 1.550000E+03 2.000000E+02 3.810000E+02
     217         0         0        11 1.550000E+03 1.000000E+02 3.810000E+02
     218         0         0        11 1.550000E+03 0.000000E+00 3.810000E+02
     219         0         0        11 1.675000E+03 2.000000E+02 3.810000E+02
     220         0         0        11 1.675000E+03 0.000000E+00 3.810000E+02
     221         0         0        11 1.800000E+03 2.000000E+02 3.810000E+02
     222         0         0        11 1.800000E+03 1.000000E+02 3.810000E+02
     223         0         0        11 1.800000E+03 0.000000E+00 3.810000E+02
     224         0         0        11 1.925000E+03 2.000000E+02 3.810000E+02
     225         0         0        11 1.925000E+03 0.000000E+00 3.810000E+02
     226         0         0        11 2.050000E+03 2.000000E+02 3.810000E+02
     227         0         0        11 2.050000E+03 1.000000E+02 3.810000E+02
     228         0         0        11 2.050000E+03 0.000000E+00 3.810000E+02
-1
-1
71
       1         6        95         1         1         7         8
       1         2         3         5         8         7         6         4
       2         6        95         1         1         7         8
       6         7         8        10        13        12        11         9
       3         6        95         1         1         7         8
      11        12        13        15        18        17        16        14

       4         6        95         1         1         7         8
      16        17        18        20        23        22        21        19
       5         6        95         1         1         7         8
      21        22        23        25        28        27        26        24
       6         6        95         1         1         7         8
      26        27        28        30        33        32        31        29
       7         6        95         1         1         7         8
      31        32        33        35        38        37        36        34
       8         6        95         1         1         7         8
      41        43        46        45        44        42        39        40
       9         6        95         1         1         7         8
      46        48        51        50        49        47        44        45
      10         6        95         1         1         7         8
      51        53        56        55        54        52        49        50
      11         6        95         1         1         7         8
      56        58        61        60        59        57        54        55
      12         6        95         1         1         7         8
      61        63        66        65        64        62        59        60
      13         6        95         1         1         7         8
      66        68        71        70        69        67        64        65
      14         6        95         1         1         7         8
      71        73        76        75        74        72        69        70
      15         6        95         1         1         7         8
      77        78        79        81        84        83        82        80
      16         6        95         1         1         7         8
      82        83        84        86        89        88        87        85
      17         6        95         1         1         7         8
      87        88        89        91        94        93        92        90
      18         6        95         1         1         7         8
      92        93        94        96        99        98        97        95
```

```
        19         6        95         1         1         7         8
        97        98        99       101       104       103       102       100
        20         6        95         1         1         7         8
       102       103       104       106       109       108       107       105
        21         6        95         1         1         7         8
       107       108       109       111       114       113       112       110
        22         6        95         1         1         7         8
       117       119       122       121       120       118       115       116
        23         6        95         1         1         7         8
       122       124       127       126       125       123       120       121
        24         6        95         1         1         7         8
       127       129       132       131       130       128       125       126
        25         6        95         1         1         7         8
       132       134       137       136       135       133       130       131
        26         6        95         1         1         7         8
       137       139       142       141       140       138       135       136
        27         6        95         1         1         7         8
       142       144       147       146       145       143       140       141
        28         6        95         1         1         7         8
       147       149       152       151       150       148       145       146
        29         6        95         1         1         7         8
       153       154       155       157       160       159       158       156
        30         6        95         1         1         7         8
       158       159       160       162       165       164       163       161
        31         6        95         1         1         7         8
       163       164       165       167       170       169       168       166
        32         6        95         1         1         7         8
       168       169       170       172       175       174       173       171
        33         6        95         1         1         7         8
       173       174       175       177       180       179       178       176
        34         6        95         1         1         7         8
       178       179       180       182       185       184       183       181
        35         6        95         1         1         7         8
       183       184       185       187       190       189       188       186
        36         6        95         1         1         7         8
       193       195       198       197       196       194       191       192
        37         6        95         1         1         7         8
       198       200       203       202       201       199       196       197
        38         6        95         1         1         7         8
       203       205       208       207       206       204       201       202
        39         6        95         1         1         7         8
       208       210       213       212       211       209       206       207
        40         6        95         1         1         7         8
       213       215       218       217       216       214       211       212
        41         6        95         1         1         7         8
       218       220       223       222       221       219       216       217
        42         6        95         1         1         7         8
       223       225       228       227       226       224       221       222
    -1
    -1

   748
         1        95
0.000000E+00 0.000000E+00 0.000000E+00
         2        95
0.000000E+00 0.000000E+00 0.000000E+00
         3        95
0.000000E+00 0.000000E+00 0.000000E+00
         4        95
0.000000E+00 0.000000E+00 0.000000E+00
         5        95
0.000000E+00 0.000000E+00 0.000000E+00
         6        95
```

```
0.000000E+00 0.000000E+00 0.000000E+00
       7        95
0.000000E+00 0.000000E+00 0.000000E+00
       8        95

0.000000E+00 0.000000E+00 0.000000E+00
       9        95
0.000000E+00 0.000000E+00 0.000000E+00
      10        95
0.000000E+00 0.000000E+00 0.000000E+00
      11        95
0.000000E+00 0.000000E+00 0.000000E+00
      12        95
0.000000E+00 0.000000E+00 0.000000E+00
      13        95
0.000000E+00 0.000000E+00 0.000000E+00
      14        95
0.000000E+00 0.000000E+00 0.000000E+00
      15        95
0.000000E+00 0.000000E+00 0.000000E+00
      16        95
0.000000E+00 0.000000E+00 0.000000E+00
      17        95
0.000000E+00 0.000000E+00 0.000000E+00
      18        95
0.000000E+00 0.000000E+00 0.000000E+00
      19        95
0.000000E+00 0.000000E+00 0.000000E+00
      20        95
0.000000E+00 0.000000E+00 0.000000E+00
      21        95
0.000000E+00 0.000000E+00 0.000000E+00
      22        95
0.000000E+00 0.000000E+00 0.000000E+00
      23        95
0.000000E+00 0.000000E+00 0.000000E+00
      24        95
0.000000E+00 0.000000E+00 0.000000E+00
      25        95
0.000000E+00 0.000000E+00 0.000000E+00
      26        95
0.000000E+00 0.000000E+00 0.000000E+00
      27        95
0.000000E+00 0.000000E+00 0.000000E+00
      28        95
0.000000E+00 0.000000E+00 0.000000E+00
      29        95
0.000000E+00 0.000000E+00 0.000000E+00
      30        95
0.000000E+00 0.000000E+00 0.000000E+00
      31        95
0.000000E+00 0.000000E+00 0.000000E+00
      32        95
0.000000E+00 0.000000E+00 0.000000E+00
      33        95
0.000000E+00 0.000000E+00 0.000000E+00
      34        95
0.000000E+00 0.000000E+00 0.000000E+00
      35        95
0.000000E+00 0.000000E+00 0.000000E+00
      36        95
0.000000E+00 0.000000E+00 0.000000E+00
      37        95
```

```
 0.000000E+00 0.000000E+00 0.000000E+00
       38        95
 0.000000E+00 0.000000E+00 0.000000E+00
       39        95
 0.000000E+00 0.000000E+00 0.000000E+00
       40        95
 0.000000E+00 0.000000E+00 0.000000E+00
       41        95
 0.000000E+00 0.000000E+00 0.000000E+00
       42        95
 0.000000E+00 0.000000E+00 0.000000E+00
   -1
   -1
  755
        1         1
RESTRAINT SET 1
        2         4 0 1 0 1 0 1 0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
        6         4 0 0 1 0 0 1 0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
        7         4 0 1 1 1 0 1 0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
        8         4 0 0 1 0 0 1 0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
       12         4 0 1 0 1 0 1 0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
       17         4 0 1 0 1 0 1 0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
       22         4 0 1 0 1 0 1 0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
       27         4 0 1 0 1 0 1 0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
       32         4 0 1 0 1 0 1 0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
       36         4 1 0 0 0 1 1 0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
       37         4 1 1 0 1 1 1 0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
       38         4 1 0 0 0 1 1 0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00


       40         4 0 1 0 1 0 1 0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
       44         4 0 0 1 0 0 1 0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
       45         4 0 1 1 1 0 1 0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
       46         4 0 0 1 0 0 1 0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
       50         4 0 1 0 1 0 1 0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
       55         4 0 1 0 1 0 1 0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
       60         4 0 1 0 1 0 1 0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
       65         4 0 1 0 1 0 1 0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
       70         4 0 1 0 1 0 1 0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
       74         4 1 0 0 0 1 1 0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
       75         4 1 1 0 1 1 1 0
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
```

```
     76        4 1 0 0 0 1 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
     78        4 0 1 0 1 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
     82        4 0 0 1 0 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
     83        4 0 1 1 1 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
     84        4 0 0 1 0 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
     88        4 0 1 0 1 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
     93        4 0 1 0 1 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
     98        4 0 1 0 1 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    103        4 0 1 0 1 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    108        4 0 1 0 1 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    112        4 1 0 0 0 1 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    113        4 1 1 0 1 1 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    114        4 1 0 0 0 1 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    116        4 0 1 0 1 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    120        4 0 0 1 0 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    121        4 0 1 1 1 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    122        4 0 0 1 0 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    126        4 0 1 0 1 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    131        4 0 1 0 1 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    136        4 0 1 0 1 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    141        4 0 1 0 1 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    146        4 0 1 0 1 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    150        4 1 0 0 0 1 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    151        4 1 1 0 1 1 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    152        4 1 0 0 0 1 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    154        4 0 1 0 1 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    158        4 0 0 1 0 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    159        4 0 1 1 1 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    160        4 0 0 1 0 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    164        4 0 1 0 1 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    169        4 0 1 0 1 0 1 0
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    174        4 0 1 0 1 0 1 0
```

```
0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    179        4 0 1 0 1 0 1 0
0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    184        4 0 1 0 1 0 1 0
0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    188        4 1 0 0 0 1 1 0
0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    189        4 1 1 0 1 1 1 0
0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    190        4 1 0 0 0 1 1 0
0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    192        4 0 1 0 1 0 1 0
0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    196        4 0 0 1 0 0 1 0
0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    197        4 0 1 1 1 0 1 0
0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    198        4 0 0 1 0 0 1 0
0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    202        4 0 1 0 1 0 1 0
0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    207        4 0 1 0 1 0 1 0
0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    212        4 0 1 0 1 0 1 0
0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    217        4 0 1 0 1 0 1 0
0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    222        4 0 1 0 1 0 1 0
0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    226        4 1 0 0 0 1 1 0
0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    227        4 1 1 0 1 1 1 0
0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
    228        4 1 0 0 0 1 1 0
0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  -1
  -1
 735
         3         1         7         3
         3         5         8
         3         2         7         9
         8        10        13        15        18        20        23        25
        28
         3         3         7         5
        28        30        33        35        38
         3         4         7         3
        36        37        38
         3         5         7         5
        26        29        31        34        36
         3         6         7         9
         6         9        11        14        16        19        21        24
        26
         3         7         7         3
         1         4         6
         3         8         7         3
         1         2         3
         3         9         7         3
         6         7         8
         3        10         7         3
        26        27        28
         5         1         7         8
         1         2         3         4         5         6         7         8
         5         1         8         1
```

```
      1
      5       2       7      23
      6       7       8       9      10      11      12      13
     14      15      16      17      18      19      20      21
     22      23      24      25      26      27      28
      5       2       8       4
      2       3       4       5
      5       3       7      13
     26      27      28      29      30      31      32      33
     34      35      36      37      38
      5       3       8       2
      6       7
 -1
```

Appendix D

Program Listing

```
C     INCLUDE (PROCESS)
      PROGRAM NLRC3D
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C INITAL    ERRFIL    INPUT     REVIEW    GLOB1     DIAGNL
C LOAD      CNTRL1    CFILE
C
C*************************************************************
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC1,MNIT,IFLAG1,IFLAG2,IDIM,
     1              NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC(10),LDCONT,INCPTR
      COMMON/INPUTG/IFLAG3,IOINTR,IFPLOT
      COMMON/CONTR1/INCREM,NIT
      COMMON/SAFE1/SKG(2000000),SKGL(20000)
C     COMMON/SAFE1/SKG(18000000),SKGL(18000000)
      COMMON/SAFE2/R(60000),IDOF(60000),JDIAG(60000)
      COMMON/IALGTM/IFLAG5
      CALL INITAL
C
C ----- IOUT = OUTPUT DEVICE NUMBER
C
      IOUT = 13
C
C ----- OPEN THE ERROR MESSAGES FILE
C
      CALL ERRFIL
C
C ----- READ THE INPUT FILE
C
      CALL INPUT (IOUT, IERROR)
C
      CALL REVIEW (IDOF)
C
C ----- DEFINE THE GLOBAL DEGREES OF FREEDOM
C ----- FIND THE BANDWIDTH AND THE LOCATION OF THE DIAGNAL TERMS
C         IN THE GLOBAL STIFFNESS MATRIX
C
      IF (IFLAG3 .EQ. 0) THEN
          CALL GLOB1 (NNODES, NNDF, NTDF, IDOF)
          CALL DIAGNL (NELEM, NNDF, NTDF, IDOF, JDIAG, NTSK, NNODES,
     1        IFLAG2, IOUT)
      ENDIF
C
C ----- ASSEMBLE THE LOAD VECTOR
C
      CALL LOAD (R, IERROR)
      IF (IFLAG5 .EQ. 0) CALL CNTRL1 (SKG, SKGL, R, IDOF, JDIAG, NTSK,
     1    NTDF, IOUT, MBAND)
  100 CONTINUE
```

```
      CALL CFILE
C
  110 CONTINUE
      STOP
      END
C
C ======================= A S S E M B ==============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE ASSEMB(SKG,SKGL,R,U,IDOF,JDIAG,NTSK,MBAND,IOUT)
C
C =================================================================
C I
C I   P R O G R A M
C I
C I   SUBROUTINE ASSEMB ASSEMBELS THE GLOBAL STIFFNESS MATRIX AND/OR
C I   STORES THE NODE NUMBERS OF THE CORENT ELEMENT AND THE POSITION
C I   OF THE ELEMENT MATRICES IN THE GLOBAL MATRICES.
C I
C I
C I   A R G U M E N T    L I S T
C I
C I   SKG(I)   =  GLOBAL STIFFNESS MATRIX STORED IN A ONE DIMENSIONAL
C I               ARRAY USING THE SKYLINE METHOD
C I   R(I)     =  LOAD VECTOR
C I   U(I)     =  VECTOR OF THE IMPOSED NODAL DISPLACEMENTS
C I   IDOF(I)  =  VECTOR CONTAINING THE D.O.F. NUMBERS THE JOINTS
C I   JDIAG(I) =  LOCATION OF THE DIAGNAL TERMS OF EACH COLUMN IN THE
C I               GLOBAL STIFFNESS MATRIX 'SKG'
C I   NTSK     =  NUMBER OF TERMS IN THE SKG MATRIX
C I   IFLAG2   =  0; SYMMETRIC STIFFNESS MATRIX
C I               1; NONSYMMETRIC STIFFNESS MATRIX
C I   IERROR   =  ERROR CODE    GE 0 ; ERRORS DETECTED
C I                             =0 ; NO ERRORS
C I
C =================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ELINFO    ELINTM    LYINFO    ISH3DG    ISHSHL    ES3DLS
C ESSHL     ZEROSK
C
C************************************************************
      INTEGER ELNUM,ELNUMB
      REAL*4 THICK
      REAL*8 LTHICK
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1             NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC1(10),LDCONT,INCPTR
      COMMON/INPUT9/THICK(9),IFLAG
      COMMON/INPUT2/NOP(20,5000)
      COMMON/ELST1/SK(60,60)
      COMMON/ASSEM2/II(120)
      COMMON/INPUTF/MATYPE(10)
      COMMON/MPCS/COEFMP(40000),ALAMB(40000),MPCDOF(40000),
     $           MPCADR(2,5000),NMPC,MPCPNT,MAXMPC
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
```

```
     $,IHIST2
      COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
      DIMENSION IDOF( 1 ),JDIAG( 1 ),SKG( 1 ),SKGL( 1 ),R( 1 ),U( 1 )
      DIMENSION  DUSK(60,60)
      IF (IFLAG2 .EQ. 0) THEN
C*VDIR: PREFER VECTOR
          DO 100 K1 = 1, NTSK
              SKG(K1) = 0.
  100     CONTINUE
      ELSE
C*VDIR: PREFER VECTOR
          DO 110 K1 = 1, NTSK/2
              SKG(K1) = 0.
              SKGL(K1) = 0.
  110     CONTINUE
      ENDIF
C
C THE STIFFNESS AND LOAD VECTORS ARE MODIFIED TO ENFORCE CONSTRAINT
C RELATIONS EXISTING BETWEEN DIFFERENT DOFS.
C
      IF (NMPC .GT. 0) THEN
          MDF = NNDF*NNODES
          DO 120 K1 = 1, NMPC
              R(IDOF(K1+MDF)) = 0.0
  120     CONTINUE
C
          DO 140 K1 = 1, NMPC
              NDOF = IDOF(MDF+K1)
              ILOC = MPCADR(1,K1)
              LOCD = JDIAG(NDOF)
              DO 130 K2 = ILOC, ILOC + MPCADR(2,K1) - 1
                  ID = MPCDOF(K2)
                  IF (ID .GT. 0) THEN
                      JDOF = IDOF(ID)
                      IF (JDOF .GT. 0) THEN
                          IF (IFLAG2 .EQ. 0) THEN
                              LOCA = LOCD + NDOF - JDOF
                              SKG(LOCA) = COEFMP(K2)
                          ELSE IF (IFLAG2 .EQ. 1) THEN
                              LOCA = LOCD - NDOF + JDOF
                              SKG(LOCA) = COEFMP(K2)
                              SKGL(LOCA) = COEFMP(K2)
                          ENDIF
                      ELSE IF (JDOF .LT. 0) THEN
                          R(NDOF) = R(NDOF) - COEFMP(K2)*U(ID)
                      ENDIF
                  ELSE IF (ID .EQ. 0) THEN
                      R(NDOF) = R(NDOF) - COEFMP(K2)
                  ENDIF
  130         CONTINUE
  140     CONTINUE
      ENDIF
C
C           NCB = NUMBER OF COLUMNS IN THE B MATRIX.
C           NRB = NUMBER OF ROWS IN THE B MATRIX.
C           NNEL = NUMBER OF NODES IN THE ELEMENT.
C           MBAN = FULL BANDWIDTH OF THE STIFFNESS MATRIX
C
      MBAN = MBAND*2 - 1
      NENDF = 0
      REWIND IHISTY
      REWIND IHIST1
      DO 280 ELNUM = 1, NELEM
```

```
C
         CALL ELINFO (ELNUM, ITYPE, NNEL, IFLAG, ISTART, LINES)
         CALL ELINTM (ELNUM, IDENT, INTCOD, NIPXI, NIPETA, NIPSI,
     1        MATNUM, THICK)
         DO IBG = 1, 60
            DO IMG = 1, 60
               DUSK(IBG,IMG) = 0.0
            END DO
         END DO
C
C*VDIR: PREFER SCALAR
C
         DO 180 K1 = 1, NNEL
            I1 = NNDF*(K1-1)
            I2 = NNDF*(NOP(K1,ELNUM)-1)
C
C*VDIR: PREFER SCALAR
C
            DO 170 K2 = 1, NNDF
               K = I1 + K2
               II(K) = I2 + K2
  170       CONTINUE
  180    CONTINUE
C
C -- NLAYRS HAVE TO BE STORED PROPERLY FOR EACH ELEMENT.
C
         DO 210 LYNUM = 1, NLAYRS
C
            CALL LYINFO (LYNUM, LTHICK, ZS, MATRL, DCS, NNEL, ELNUM,
     1           NIPXI, NIPETA, NIPSI)
C
C --- THE D MATRIX OF CONCRETE/STEEL LAYER IS RECOVERED HERE.
C
            NIP = NIPXI*NIPETA*NIPSI
            IF (MATYPE(MATNUM) .EQ. 4) THEN
               READ (IHISTY) ELNUMB, LYNUMB, NGAUS, ((CRACK(L,K,LYNUM
     1            ),K = 1,250), L = 1, NIP)
               IF (ELNUM.NE.ELNUMB .OR. LYNUM.NE.LYNUMB .OR. NIP.NE.
     1            NGAUS) THEN
                  WRITE (*, *) 'READ ERROR READING IHISTY IN ASSEMB'
                  WRITE (*, *) 'ELNUM', ELNUM, 'ELNUMB', ELNUMB,
     1               'LYNUM', LYNUM, 'LYNUMB', LYNUMB, 'NIP', NIP,
     2               'NGAUS', NGAUS
                  STOP
               ENDIF
            ENDIF
C
C ----- NENDF = NUMBER OF ELEMENT NODAL DEGREES OF FREEDOM
C ----- NCB   = NUMBER OF COLUMNS IN THE B MATRIX
C ----- NRB   = NUMBER OF ROWS IN THE B MATRIX
C ----- ITYPE = ELEMENT TYPE
C ----- IFLAG = ADDITIONAL IDENTIFIER FOR THE ELEMENT
C
            IF (ITYPE .LE. 0) GO TO 270
            IF (ITYPE.NE.0 .OR. IDENT.NE.0) THEN
               IF (ITYPE .GT. 300) THEN
                  IF (IFLAG .EQ. 0) THEN
                     NCB = 3*NNEL
                     NRB = 6
                     NENDF = 3
                     CALL ISH3DG (ITYPE, NNEL, IERROR)
                  ELSE IF (IFLAG .EQ. 4) THEN
                     NCB = 6*NNEL
```

```
                    NRB = 6
                    NENDF = 6
                    CALL ISHSHL (ITYPE, NNEL, IERROR)
                ENDIF
            ENDIF
        ENDIF
C
C ----- GEOMETRICALLY LINEAR PROBLEMS
C
        IF (IFLAG1 .EQ. 0) THEN
            IF (ITYPE .GT. 300) THEN
                IF (IFLAG .EQ. 0) THEN
                    CALL ES3DLS (ELNUM, ITYPE, NNEL, NNDF, NRB,
     1                   NCB, NIP, MATNUM, IFLAG2, IOUT)
                ELSE IF (IFLAG .EQ. 4) THEN
                    CALL ESSHL (ELNUM, ITYPE, NNEL, NNDF, NRB, NCB
     1                  , NIP, MATNUM, IFLAG2, IOUT)
                ENDIF
            ENDIF
        ENDIF
        DO IBG = 1, NCB
            DO IMG = 1, NCB
                DUSK(IBG,IMG) = DUSK(IBG,IMG) + SK(IBG,IMG)
            END DO
        END DO
  210   CONTINUE
        CALL ZEROSK (NCB)
        DO IMG = 1, NCB
            DO IBG = 1, NCB
                SK(IMG,IBG) = DUSK(IMG,IBG)
            END DO
        END DO
C
C ----- NDIF = DIFFERENCE BETWEEN THE GLOBAL NODAL DEGREES OF FREEDOM
C              AND THE ELEMENT NODAL DEGREES OF FREEDOM
C
        NDIF = NNDF - NENDF
        DO 260 K1 = 1, NCB
            ID1 = (K1-1)/NENDF
            ID1 = K1 + ID1*NDIF
            NDOF = IDOF(II(ID1))
            IF (NDOF .GT. 0) THEN
                LOCD = JDIAG(NDOF)
                DO 240 K2 = 1, NCB
                    ID2 = (K2-1)/NENDF
                    JDOF = IDOF(II(K2+ID2*NDIF))
                    IF (IFLAG2 .EQ. 0) THEN
                        IF (NDOF.GE.JDOF .AND. JDOF.GT.0) THEN
                            LOCA = LOCD + NDOF - JDOF
                            SKG(LOCA) = SKG(LOCA) + SK(K1,K2)
                        ENDIF
                    ELSE
                        IF (NDOF.GE.JDOF .AND. JDOF.GT.0) THEN
                            LOCA = LOCD - NDOF + JDOF
                            SKG(LOCA) = SKG(LOCA) + SK(K2,K1)
                            SKGL(LOCA) = SKGL(LOCA) + SK(K1,K2)
                        ENDIF
                    ENDIF
  240           CONTINUE
            ELSE IF (NDOF .LT. 0) THEN
                DO 250 K2 = 1, NCB
                    ID2 = (K2-1)/NENDF
                    JDOF = IDOF(II(K2+ID2*NDIF))
```

```
                  IF (JDOF .GT. 0) R(JDOF) = R(JDOF) - SK(K1,K2)*U(
    1                II(ID1))
  250           CONTINUE
              ENDIF
  260     CONTINUE
  270     CONTINUE
  280 CONTINUE
  290 CONTINUE
      RETURN
      END

C
C ======================== E L S T I F ==========================
C
C     INCLUDE (PROCESS)
      SUBROUTINE ELSTIF
C
C ===============================================================
C I
C I   P R O G R A M
C I
C I   ELSTIF EVALUATES THE STIFFNESS MATIRIX OF EACH ELEM.
C I
C I   E N T R Y   P O I N T S
C I
C I
C I   ES3DLS: FOR 3D STRAIN FIELDS WITHOUT GEOM. NONLINEARITY
C I
C I   ESSHL : FOR 3D STRAIN FIELDS WITHOUT GEOM. NONLINEARITY
C I
C I
C I
C I   A R G U M E N T   L I S T
C I
C I   ELNUM     = ELEMENT NUMBER
C I
C I   NNEL      = NUMBER OF NODES IN THE ELEMENT
C I
C I   NRB       = NUMBER OF ROWS OF THE B MATRIX
C I
C I   NCB       = NUMBER OF COLUMNS OF THE B MATRIX
C I
C I   NIP       = TOTAL NUMBER OF INTEGRATION POINTS IN THE ELEM.
C I
C I   MATNUM    = MATERIAL NUMBER FOR THE ELEMENT
C I
C I   IERROR    = ERROR CODE
C I
C I
C ===============================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ZEROSK    JACB3D    B3DLS     MATMOD    BTDB      SKTRAN
C SHNORM    DIRVEC    GETTHK    JACSHL    BSHL      SKTSHL
C
C*********************************************************
      REAL*4 SHELLZ,THICK,XYZ
```

```
      REAL*8 N,NXI,NETA,NSI,NX,NY,NZ,SI
      CHARACTER*153 DUMMY
      INTEGER ELNUM
      COMMON/INPUTE/ISPB(10000)
      COMMON/SKTR2/SHELLZ(3,10000)
      COMMON/UTIL1/STRESS(6),DUMMY
      COMMON/INPUT9/THICK(9),IFLAG
      COMMON/SHLDIR/VECT(3,3,9)
      COMMON/ISHAP2/W(27)
      COMMON/TRANS/V1(3),V2(3),V3(3)
      COMMON/ELST1/SK(60,60)
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/ISHAP1/N(20,27),NXI(20,27),NETA(20,27),NSI(20,27),SI(27)
      COMMON/JACOB1/NX(20),NY(20),NZ(20),SIX,SIY,SIZ
      COMMON/INPUT3/XYZ(3,10000)
      COMMON/INPUT2/NOP(20,5000)
C
      ENTRY ES3DLS(ELNUM,ITYPE,NNEL,NNDF,NRB,NCB,NIP,MATNUM,IFLAG2,IOUT)
C
      CALL ZEROSK (NCB)
      DO 100 INTGPN = 1, NIP
          CALL JACB3D (INTGPN, ELNUM, NNEL, IERROR, DETJAC)
          CALL B3DLS (NNEL)
          CALL MATMOD (ELNUM, ITYPE, MATNUM, INTGPN, IFLAG, IOUT, DETJAC
     1         , 0, NNEL)
          CST = DETJAC*W(INTGPN)
          CALL BTDB (IFLAG2, NRB, NCB, CST)


  100 CONTINUE
      CALL SKTRAN (ELNUM, NNEL, 3, NCB, 3, 1)
C
      RETURN
C
      ENTRY ESSHL (ELNUM, ITYPE, NNEL, NNDF, NRB, NCB, NIP, MATNUM,
     1    IFLAG2, IOUT)
C
C
C ---- EVALUATE THE LOCAL SHELL COORDINATES OF THE NODES
C
      CALL ZEROSK (NCB)
      DO 110 K1 = 1, NNEL
          KP = NOP(K1,ELNUM)
          ICODE = IAND(ISPB(KP),4)
C
C ---- iF SHELL ROTATIONS ARE ASSEMBLED IN THE LOCAL SHELL COORDINATE
C       SYSTEM THEN RETREIVE THE LOCAL z-AXIS FROM STORAGE. ELSE
C       EVALUATE THE NORMAL TO THE SHELL MID PLANE BY A CALL TO THE
C       SHNORM ROUTINE.
C
          IF (ICODE .GT. 0) THEN
              CALL SHNORM (ELNUM, K1, NNEL, ITYPE, VECT(1,1,K1),VECT(1,2
     1             ,K1),VECT(1,3,K1))
          ELSE
              VECT(1,3,K1) = DBLE(SHELLZ(1,KP))
              VECT(2,3,K1) = DBLE(SHELLZ(2,KP))
              VECT(3,3,K1) = DBLE(SHELLZ(3,KP))
              CALL DIRVEC (VECT(1,1,K1),VECT(1,2,K1),VECT(1,3,K1))
          ENDIF
  110 CONTINUE
C
      CALL ZEROSK (NCB)
      DO 120 INTGPN = 1, NIP
          CALL GETTHK (INTGPN, ELNUM, NNEL, THICKE, RAD)
```

```
C
C ----- vector V3 returned by JACSHL is normal to mid surface of the
C       shell at integration points
C
         ISET = 0
         SIP = SI(INTGPN)
         CALL JACSHL(INTGPN,ELNUM,NNEL,THICKE,DETJAC,V3,ISET,SIP)
C
C ----- THE COORDINATES VECTORS V1 AND V2 ARE EVALUATED BY DIRVEC
C       WHICH IS PART OF THE eLEMENT lIBRARY mODULE
C
         CALL DIRVEC (V1, V2, V3)

         CALL BSHL (ELNUM, NNEL, INTGPN, THICKE)
         CALL MATMOD (ELNUM, ITYPE, MATNUM, INTGPN, IFLAG, IOUT, DETJAC
     1        , 0, NNEL)
         CST = DETJAC*W(INTGPN)
         CALL BTDB (IFLAG2, NRB, NCB, CST)
  120 CONTINUE
      CALL SKTSHL (ELNUM, NNEL, 6, NCB, 3, 2)
      CALL SKTRAN (ELNUM, NNEL, 6, NCB, 3, 3)
C
      RETURN
      END
C
C ====================== Z E R O S K ==============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE ZEROSK(N)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C************************************************************
      REAL*8 SK
      COMMON/ELST1/SK(60,60)
C
C*VDIR: PREFER VECTOR
C
      DO 110 K1 = 1, N
          DO 100 K2 = 1, N
              SK(K1,K2) = 0.
  100     CONTINUE
  110 CONTINUE
C
      RETURN
      END
C
C ========================= S K T R A N ==============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE SKTRAN(ELNUM,NNEL,NENDF,NCB,IDIM,ICASE)
C
C ===============================================================
C I
C I   P R O G R A M:
C I
C I   SKTRAN MODIFIES THE ELEMENT STIFFNESS MATRIX FOR THE SKEW
```

```
C I   BOUNDARY CONDITIONS USING T tRANS  SK T TRANSFORMATION.
C I
C I   A R G U M E N T    L I S T:
C I
C I   ELNUM      = element number
C I   NNEL       = number of nodes in the element
C I   NENDF      = number of element nodal degrees of freedom
C I   NCB        = number of columns of the B matrix
C I   IDIM       = physical dimension of the problem (i.e.,2D or 3D)
C I   ICASE      = special code
C I                1; transform displacement components
C I                2; transform rotation components
C I                3; transform all components
C I
C I
C ==================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C DIRCOS
C
C************************************************************
      INTEGER ELNUM
      COMMON/ELST1/SK(60,60)
      COMMON/TRANS/DC(3,3)
      COMMON/SHLDIR/VECT(3,3,9)
      COMMON/B1/B(6,60)
      COMMON/INPUTE/ISPB(10000)
      COMMON/INPUT2/NOP(20,5000)
      DIMENSION CST(60,3)
C
      IS = 0
      IE = 0
      IF (ICASE .EQ. 1) THEN
          IE = 1
          IS = 1
      ELSE IF (ICASE .EQ. 2) THEN
          IS = 2
      ELSE IF (ICASE .EQ. 3) THEN
          IS = 1
          IE = 2
      ENDIF
C
      DO 190 K1 = 1, NNEL
          NODE = NOP(K1,ELNUM)
          DO 180 IC = IS, IE
              ICODE = IAND(ISPB(NODE),IC)
              IF (ICODE .GT. 0) THEN
                  CALL DIRCOS (ICODE, IDIM, NODE)
                  I = NENDF*(K1-1) + IDIM*(IC-1)
C
C*VDIR: PREFER VECTOR
C
                  DO 130 K2 = 1, NCB
                      DO 110 K3 = 1, IDIM
                          CST(K2,K3) = 0.
                          DO 100 IDIR = 1, IDIM
                              ID = I + IDIR
```

```
                     CST(K2,K3) = CST(K2,K3) + SK(K2,ID)*DC(
     1                   IDIR,K3)
  100                CONTINUE
  110             CONTINUE
C
                  DO 120 K3 = 1, IDIM
                     ID = I + K3
                     SK(K2,ID) = CST(K2,K3)
  120             CONTINUE
  130          CONTINUE
C
C*VDIR: PREFER VECTOR
               DO 170 K2 = 1, NCB
                  DO 150 K3 = 1, IDIM
                     CST(K2,K3) = 0.
C
                     DO 140 IDIR = 1, IDIM
                        ID = I + IDIR
                        CST(K2,K3) = CST(K2,K3) + SK(ID,K2)*DC(
     1                      IDIR,K3)
  140                CONTINUE
  150             CONTINUE
C
                  DO 160 K3 = 1, IDIM
                     ID = I + K3
                     SK(ID,K2) = CST(K2,K3)
  160             CONTINUE
  170          CONTINUE
            ENDIF
  180    CONTINUE
  190 CONTINUE
      RETURN
C
C
C
C
      ENTRY SKTSHL (ELNUM, NNEL, NENDF, NCB, IDIM, ICASE)
      IS = 0
      IE = 0
      IF (ICASE .EQ. 1) THEN
         IE = 1
         IS = 1
      ELSE IF (ICASE .EQ. 2) THEN
         IE = 2
         IS = 2
      ELSE IF (ICASE .EQ. 3) THEN
         IS = 1
         IE = 2
      ENDIF
C
      DO 290 K1 = 1, NNEL
         NODE = NOP(K1,ELNUM)
         ICODE = IAND(ISPB(NODE),4)
         IF (ICODE .GT. 0) THEN
            DO 280 IC = IS, IE
               I = NENDF*(K1-1) + IDIM*(IC-1)
C
C*VDIR: PREFER VECTOR
C
               DO 230 K2 = 1, NCB
                  DO 210 K3 = 1, IDIM
                     CST(K2,K3) = 0.
                     DO 200 IDIR = 1, IDIM
```

```
                        ID = I + IDIR
                        CST(K2,K3) = CST(K2,K3) + SK(K2,ID)*VECT(
     1                     K3,IDIR,K1)
  200                 CONTINUE
  210              CONTINUE
C
                   DO 220 K3 = 1, IDIM
                      ID = I + K3
                      SK(K2,ID) = CST(K2,K3)
  220              CONTINUE
  230           CONTINUE
C
C*VDIR: PREFER VECTOR
                DO 270 K2 = 1, NCB
                   DO 250 K3 = 1, IDIM
                      CST(K2,K3) = 0.
C
                      DO 240 IDIR = 1, IDIM
                         ID = I + IDIR
                         CST(K2,K3) = CST(K2,K3) + SK(ID,K2)*VECT(
     1                     K3,IDIR,K1)
  240                 CONTINUE
  250              CONTINUE
C
                   DO 260 K3 = 1, IDIM
                      ID = I + K3
                      SK(ID,K2) = CST(K2,K3)
  260              CONTINUE
  270           CONTINUE
  280        CONTINUE
          ENDIF
  290 CONTINUE
      RETURN
C
      ENTRY BTSHL (ELNUM, NNEL, NENDF, NRB, IDIM, ICASE)
      IS = 0
      IE = 0
      IF (ICASE .EQ. 1) THEN
          IE = 1
          IS = 1
      ELSE IF (ICASE .EQ. 2) THEN
          IE = 2
          IS = 2
      ELSE IF (ICASE .EQ. 3) THEN
          IS = 1
          IE = 2
      ENDIF
C
      DO 350 K1 = 1, NNEL
          NODE = NOP(K1,ELNUM)
          ICODE = IAND(ISPB(NODE),4)
          IF (ICODE .GT. 0) THEN
              DO 340 IC = IS, IE
                  I = NENDF*(K1-1) + IDIM*(IC-1)
C
C*VDIR: PREFER VECTOR
C
                  DO 330 K2 = 1, NRB
                      DO 310 K3 = 1, IDIM
                          CST(K2,K3) = 0.
                          DO 300 IDIR = 1, IDIM
                              ID = I + IDIR
                              CST(K2,K3) = CST(K2,K3) + B(K2,ID)*VECT(K3
```

```
    1                            ,IDIR,K1)
  300                    CONTINUE
  310                CONTINUE
C
                     DO 320 K3 = 1, IDIM
                         ID = I + K3
                         B(K2,ID) = CST(K2,K3)
  320                CONTINUE
  330            CONTINUE
C
  340        CONTINUE
         ENDIF
  350 CONTINUE
      RETURN
      END
C
C ========================== D I R C O S ===============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE DIRCOS(ICODE,IDIM,NODE)
C
C ===================================================================
C I
C I   P R O G R A M:
C I
C I   DIRCOS returns the transformation matrix which is used to
C I   define the local coordinates at a node.
C I
C I
C I   A R G U M E N T    L I S T
C I
C I   ICODE     = 1; get the transformation matrix for the
C I                  translational local coordinate.
C I               2; get the transformation matrix for the
C I                  rotational local coordinate.
C I
C I   IDIM      = physical dimension of the problem (i.e.,2D or 3D)
C I
C I   NODE      = node number of the structure
C I
C I
C I   C O M M O N    B L O C K S
C I
C I   XAXIS    = contains the first two direction cosines of the
C I              translation and rotation X axes.
C I
C I   YAXIS    = contains the first two direction cosines of the
C I              translation and rotation Y axis.
C I
C I   DC(I,J)  = transformation matrix which has its columns equal
C I              to the direction cosines of the local X, Y and Z axes
C I
C I
C ===================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C CROSS
```

```
C
C*************************************************************
      REAL*4 XAXIS,YAXIS
      LOGICAL*4 TEST1,TEST2
      COMMON/INPUTD/XAXIS(4 , 10000),YAXIS(4 , 10000)
      COMMON/INPUTE/ISPB(10000)
      COMMON/TRANS/DC(3,3)
C
      IF (ICODE .EQ. 1) THEN
          DC(1,1) = DBLE(XAXIS(1,NODE))
          DC(2,1) = DBLE(XAXIS(2,NODE))
          DC(1,2) = DBLE(YAXIS(1,NODE))
          DC(2,2) = DBLE(YAXIS(2,NODE))
      ELSE IF (ICODE .EQ. 2) THEN
          DC(1,1) = DBLE(XAXIS(3,NODE))
          DC(2,1) = DBLE(XAXIS(4,NODE))
          DC(1,2) = DBLE(YAXIS(3,NODE))
          DC(2,2) = DBLE(YAXIS(4,NODE))
      ENDIF
C
      IF (IDIM .EQ. 2) THEN
          RETURN
      ELSE IF (IDIM .EQ. 3) THEN
C
C ---- find the third direction cosine of the local X and Y axes.
C
          DC(3,1) = DSQRT(1.0D0-DC(1,1)**2+DC(2,1)**2)
          DC(3,2) = DSQRT(1.0D0-DC(1,2)**2+DC(2,2)**2)
C
C ---- Check the sign bits for the third direction cosines.
C
          TEST1 = .FALSE.
          TEST2 = .FALSE.
          IF (ICODE .EQ. 1) THEN
              TEST1 = BTEST(ISPB(NODE),3)
              TEST2 = BTEST(ISPB(NODE),4)
          ELSE IF (ICODE .EQ. 2) THEN
              TEST1 = BTEST(ISPB(NODE),5)
              TEST2 = BTEST(ISPB(NODE),6)
          ENDIF
          IF (TEST1) DC(3,1) = -DC(3,1)
          IF (TEST2) DC(3,2) = -DC(3,2)
C
C ---- take the cross product of the first two vectors to find the
C       the local Z axis.
C
          CALL CROSS (DC(1,1),DC(1,2),DC(1,3))


       ENDIF
      RETURN
      END
C
C ========================== B T D B ==============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE BTDB(IFLAG2,NRB,NCB,CST)
C
C ================================================================
C I
C I   SUBPROGRAM BTDB EVALUATES  B Trans DEP B CST, WHERE
C I
C I      BT = TRANSPOSE OF THE B MATRIX
C I      DEP = MATERIAL STIFFNESS MATRIX
```

```
C I      CST   = CONSTANT VALUE TO BE MULT. WITH EACH TERM OF THE
C I              RESULTING MATRIX.
C I
C I   A R G U M E N T     L I S T
C I
C I      IFLAG2 = 0; FOR SYMMETRIC STIFFNESS MATRIX
C I               1; FOR NONSYMMETRIC STIFFNESS MATRIX
C I
C I      NRB  = NUMBER OF ROWS IN THE B BATRIX
C I
C I      NCB  = NUMBER OF COLUMNS IN THE B MATRIX
C I
C I      CST  = INTEGRATION CONSTANT
C I
C I
C I
C ===================================================================
C
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C************************************************************
      REAL*8 SK,B,DEP,CST,DUMMY,TEMP
      COMMON/ELST1/SK(60,60)
      COMMON/B1/B(6,60)
      COMMON/MATER1/DEP(6,6)
      DIMENSION DUMMY(60,6)
C
C --- B(K3,K1) IS THE TRANSPOSE OF THE B(K1,K3)
C
      DO 110 K1 = 1, NRB
          DO 100 K2 = 1, NRB
              DEP(K1,K2) = DEP(K1,K2)*CST
  100     CONTINUE
  110 CONTINUE
C
      IF (IFLAG2 .EQ. 0) THEN
C
C*VDIR: PREFER VECTOR
C
          DO 140 K1 = 1, NCB
              DO 130 K2 = 1, NRB
                  TEMP = 0.
                  DO 120 K3 = 1, NRB
                      TEMP = TEMP + B(K3,K1)*DEP(K3,K2)
  120             CONTINUE
                  DUMMY(K1,K2) = TEMP
  130         CONTINUE
  140     CONTINUE
C
C*VDIR: PREFER VECTOR
C
          DO 170 K1 = 1, NCB
              DO 160 K2 = 1, NCB
                  TEMP = 0.
                  DO 150 K3 = 1, NRB
                      TEMP = TEMP + DUMMY(K1,K3)*B(K3,K2)
```

```
  150             CONTINUE
                  SK(K1,K2) = SK(K1,K2) + TEMP
  160         CONTINUE
  170     CONTINUE
C
      ELSE IF (IFLAG2 .EQ. 1) THEN
C
C*VDIR: PREFER VECTOR
C
          DO 220 K1 = 1, NCB
              DO 190 K2 = 1, NRB
                  DUMMY(K1,K2) = 0.
                  DO 180 K3 = 1, NRB
                      DUMMY(K1,K2) = DUMMY(K1,K2) + B(K3,K1)*DEP(K3,K2)
  180             CONTINUE
  190         CONTINUE
              DO 210 K4 = 1, NCB
                  DO 200 K5 = 1, NRB
                      SK(K1,K4) = SK(K1,K4) + DUMMY(K1,K5)*B(K5,K4)
  200             CONTINUE
  210         CONTINUE
  220     CONTINUE
      ENDIF
C
      RETURN
      END
C
C ======================== B 2 D 3 D ==============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE B2D3D
C
C ================================================================
C I
C I   SUBPROGRAM B2D3D EVALUATES THE 'B' MATRIX FOR THE 2D AND 3D
C I   FINITE STRAIN PROBLEMS.
C I
C I      ENTRY POINTS:
C I      B3DLS : FOR 3D STRAIN FIELDS WITHOUT GEOMETRIC NONLINEARITY
C I
C I      BSHLS : FOR 3D STRAIN FIELDS WITHOUT GEOMETRIC NONLINEARITY
C I              FOR SHELLS
C I
C I     B(I,J)       =  VARIATIONAL STRAIN-DISPLACEMENT STIFFNESS
C I                     MATRIX.
C I
C I
C I   NX(K) = PARTIAL DERIVATIVE OF N(K) WITH RESPECT TO X;
C I   NY(K) = PARTIAL DERIVATIVE OF N(K) WITH RESPECT TO Y;
C I   NZ(K) = PARTIAL DERIVATIVE OF N(K) WITH RESPECT TO Z;
C I
C I
C I
C ================================================================
C
      IMPLICIT REAL*8 (A-H,N-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C**********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
```

```
C
C*************************************************************
      INTEGER ELNUM,NNEL,NNDF,NLAYRS
      REAL*4 THICK
      REAL*8 N,NXI,NETA,NSI,SI,LTHICK
      COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
      COMMON/INPUT9/THICK(9),IFLAG
      COMMON/ISHAP1/N(20,27),NXI(20,27),NETA(20,27),NSI(20,27),SI(27)
      COMMON/MAIN2/UTOTAL(60000)
      COMMON/ASSEM2/II(120)
      COMMON/JACOB1/NX(20),NY(20),NZ(20),SIX,SIY,SIZ
      COMMON/SHLDIR/VECT(3,3,9)
      COMMON/B1/B(6,60)
      COMMON/B2/DUDX,DVDX,DWDX,DUDY,DVDY,DWDY,DUDZ,DVDZ,DWDZ,A5
      DIMENSION V1( 3 ),V2( 3 ),V3( 3 )
      DATA ZERO,ONE/0.0D0,1.0D0/
C
      ENTRY B3DLS(NNEL)
C
C --- CALCULATION OF THE B MATRIX
C
C*VDIR: ASSUME COUNT(20)
      DO 100 K1 = 1, NNEL
          K13 = 3*K1
          K12 = K13 - 1
          K11 = K13 - 2
          TEMP1 = NX(K1)
          B(1,K11) = TEMP1
          B(4,K12) = TEMP1
          B(6,K13) = TEMP1
          TEMP2 = NY(K1)
          B(4,K11) = TEMP2
          B(2,K12) = TEMP2
          B(5,K13) = TEMP2
          TEMP3 = NZ(K1)
          B(6,K11) = TEMP3
          B(5,K12) = TEMP3
          B(3,K13) = TEMP3
          B(1,K12) = ZERO
          B(1,K13) = ZERO
          B(2,K11) = ZERO
          B(2,K13) = ZERO
          B(3,K11) = ZERO
          B(3,K12) = ZERO
          B(4,K13) = ZERO
          B(5,K11) = ZERO
          B(6,K12) = ZERO
  100 CONTINUE
C
      RETURN
C
C
C ----- evaluate the B matrix for shell elements
C
      ENTRY BSHL (ELNUM, NNEL, INTGPN, THICKE)
C
      ASI = SI(INTGPN)
      CONST = 0.5*THICKE
      DO 110 K1 = 1, NNEL
C
C ----- get the local shell coordinates of the node
C
          V1(1) = VECT(1,1,K1)
```

```
         V2(1) = VECT(1,2,K1)
         V3(1) = VECT(1,3,K1)
         V1(2) = VECT(2,1,K1)
         V2(2) = VECT(2,2,K1)
         V3(2) = VECT(2,3,K1)
         V1(3) = VECT(3,1,K1)
         V2(3) = VECT(3,2,K1)
         V3(3) = VECT(3,3,K1)
C
C ----- compute the B matrix
C
         NSIX = N(K1,INTGPN)*SIX
         NSIY = N(K1,INTGPN)*SIY
         NSIZ = N(K1,INTGPN)*SIZ
         NXASI = NX(K1)*ASI
         NYASI = NY(K1)*ASI
         NZASI = NZ(K1)*ASI
C
         K11 = 6*(K1-1) + 1
         K12 = K11 + 1
         K13 = K12 + 1
         K14 = K13 + 1
         K15 = K14 + 1
         K16 = K15 + 1
         B(1,K11) = NX(K1)
         B(2,K11) = ZERO
         B(3,K11) = ZERO
         B(4,K11) = NY(K1)
         B(5,K11) = ZERO
         B(6,K11) = NZ(K1)
         B(1,K12) = ZERO
         B(2,K12) = NY(K1)
         B(3,K12) = ZERO
         B(4,K12) = NX(K1)
         B(5,K12) = NZ(K1)
         B(6,K12) = ZERO
         B(1,K13) = ZERO
         B(2,K13) = ZERO
         B(3,K13) = NZ(K1)
         B(4,K13) = ZERO
         B(5,K13) = NY(K1)
         B(6,K13) = NX(K1)
         B(1,K14) = -CONST*(NXASI+NSIX)*V2(1)
         B(2,K14) = -CONST*(NYASI+NSIY)*V2(2)
         B(3,K14) = -CONST*(NZASI+NSIZ)*V2(3)
         B(4,K14) = -CONST*((NYASI+NSIY)*V2(1)+(NXASI+NSIX)*V2(2))
         B(5,K14) = -CONST*((NZASI+NSIZ)*V2(2)+(NYASI+NSIY)*V2(3))
         B(6,K14) = -CONST*((NZASI+NSIZ)*V2(1)+(NXASI+NSIX)*V2(3))
         B(1,K15) = CONST*(NXASI+NSIX)*V1(1)
         B(2,K15) = CONST*(NYASI+NSIY)*V1(2)
         B(3,K15) = CONST*(NZASI+NSIZ)*V1(3)
         B(4,K15) = CONST*((NYASI+NSIY)*V1(1)+(NXASI+NSIX)*V1(2))
         B(5,K15) = CONST*((NZASI+NSIZ)*V1(2)+(NYASI+NSIY)*V1(3))
         B(6,K15) = CONST*((NZASI+NSIZ)*V1(1)+(NXASI+NSIX)*V1(3))
         B(1,K16) = ZERO
         B(2,K16) = ZERO
         B(3,K16) = ZERO
         B(4,K16) = ZERO
         B(5,K16) = ZERO
         B(6,K16) = ZERO
  110 CONTINUE
C
      RETURN
```

```
      END
C
C ========================= G E T S T R ===========================
C
C     INCLUDE (PROCESS)
      SUBROUTINE GETSTR(IOUT)
C
C ==================================================================
C I
C I   SUBROUTINE GETSTR ASSEMBELS THE GLOBAL STIFFNESS MATRIX AND/OR
C I   STORES THE NODE NUMBERS OF THE CORENT ELEMENT AND THE POSITION
C I   OF THE ELEMENT MATRICES IN THE GLOBAL MATRICES.
C I
C I
C I   II(J)     POSITION OF LOCAL STIFFNESS TERMS IN THE GLOBAL
C I             STIFFNESS MATRIX.
C I
C I   SKG(I) =    GLOBAL STIFFNESS MATRIX IN THE CONDENSED FORM
C I   SK(I,J)  =    ELEMENT STIFFNESS MATRIX
C I                 (SK IS COMPUTED BY SUBPROGRAM STIFEL)
C I
C I
C ==================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ELINFO    ELINTH    LYINFO    ISH3DG    ISHSHL    S3DLS
C SSHL
C
C************************************************************
      INTEGER ELNUM,ELNUMB
      REAL*4 THICK
      REAL*8 LTHICK
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC1,MNIT,IFLAG1,IFLAG2,IDIM,
     1             NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC(10),LDCONT,INCPTR
      COMMON/INPUT9/THICK(9),IFLAG
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/INPUT2/NOP(20,5000)
      COMMON/ASSEM2/II(120)
      COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
      COMMON/INPUTF/MATYPE(10)
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON/MAIN4/RE(60000)
      COMMON/SAFE2/R(60000),IDOF(60000),JDIAG(60000)
      COMMON/MPCS/COEFMP(40000),ALAMB(40000),MPCDOF(40000),
     $           MPCADR(2,5000),NMPC,MPCPNT,MAXMPC
C
C ----- NCB = number of columns in the B matrix
C ----- NRB = number of rows in the B matrix
C ----- NNEL = number of nodes in the element
C
      REWIND IHISTY
      REWIND IHIST1
      DO 140 ELNUM = 1, NELEM
C
```

```
         CALL ELINFO (ELNUM, ITYPE, NNEL, IFLAG, ISTART, LINES)
         CALL ELINTM (ELNUM, IDENT, INTCOD, NIPXI, NIPETA, NIPSI,
     1       MATNUM, THICK)
C
C*VDIR: PREFER SCALAR
         DO 110 K1 = 1, NNEL
             I1 = NNDF*(K1-1)
             I2 = NNDF*(NOP(K1,ELNUM)-1)
C
C*VDIR: PREFER SCALAR
             DO 100 K2 = 1, NNDF
                 K = I1 + K2
                 II(K) = I2 + K2
  100        CONTINUE
  110    CONTINUE
C
         DO 120 LYNUM = 1, NLAYRS
C
             CALL LYINFO (LYNUM, LTHICK, ZS, MATRL, DCS, NNEL, ELNUM,
     1           NIPXI, NIPETA, NIPSI)
C
             NIP = NIPXI*NIPETA*NIPSI
             IF (MATYPE(MATNUM) .EQ. 4) THEN
                 READ (IHISTY) ELNUMB, LYNUMB, NGAUS, ((CRACK(L,K,LYNUM
     1               ),K = 1,250), L = 1, NIP)
                 IF (ELNUM.NE.ELNUMB .OR. LYNUM.NE.LYNUMB .OR. NIP.NE.
     1               NGAUS) THEN
                     WRITE (*, *) 'READ ERROR READING IHISTY IN GETSTR'
                     WRITE (*, *) 'ELNUM', ELNUM, 'ELNUMB', ELNUMB,
     1                   'LYNUM', LYNUM, 'LYNUMB', LYNUMB, 'NIP', NIP,
     2                   'NGAUS', NGAUS
                     STOP
                 ENDIF
             ENDIF
C
C ----- NENDF = number of element nodal degrees of freedom
C ----- NCB   = number of columns in the B matrix
C ----- NRB   = number of rows in the B matrix
C ----- ITYPE = element type
C ----- IFLAG = additional identifier for the element
C
             IF (ITYPE .LE. 0) GO TO 130
             IF (ITYPE.NE.0 .OR. IDENT.NE.0) THEN
                 IF (ITYPE .GT. 300) THEN
                     IF (IFLAG .EQ. 0) THEN
                         NCB = 3*NNEL
                         NRB = 6
                         NENDF = 3
                         CALL ISH3DG (ITYPE, NNEL, IERROR)
                     ELSE IF (IFLAG .EQ. 4) THEN
                         NCB = 6*NNEL
                         NRB = 6
                         NENDF = 6
                         CALL ISHSHL (ITYPE, NNEL, IERROR)
                     ENDIF
                 ENDIF
             ENDIF
C
C             GEOMETRICALLY LINEAR PROBLEMS
C
             IF (IFLAG1 .EQ. 0) THEN
                 IF (ITYPE .GT. 300) THEN
                     IF (IFLAG .EQ. 0) THEN
```

```
                        CALL S3DLS (ELNUM, ITYPE, NNEL, NRB, NCB, NIP
     1                       , MATNUM, NNDF, NENDF, IOUT)
                    ELSE IF (IFLAG .EQ. 4) THEN
                        CALL SSHL (ELNUM, ITYPE, NNEL, NRB, NCB, NIP,

     1                       MATNUM, NNDF, NENDF, IDIM, IOUT)
                    ENDIF
                ENDIF
            ENDIF
  120     CONTINUE
  130     CONTINUE
  140 CONTINUE
C
      IF (NMPC .GT. 0) THEN
          DO IMPC = 1, NMPC
              ILOC = MPCADR(1,IMPC)
              ICOUNT = MPCADR(2,NMPC)
              DO ICP = ILOC + 1, ILOC + ICOUNT - 1
                  IDENT = MPCDOF(ICP)
                  FACTOR = COEFMP(ICP)
                  RE(IDENT) = RE(IDENT) - ALAMB(IMPC)*FACTOR
              END DO
          END DO
      ENDIF
C
  170 CONTINUE
      RETURN
      END
C
C ======================== E L S T R ==============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE ELSTR
C
C =================================================================
C I
C I   SUBPROGRAM ELSTR EVALUATES THE STIFFNESS MATIRIX OF EACH ELEM.
C I
C I      ENTRY POINTS:
C I      S3DLS   : FOR 3D STRAIN FIELDS WITHOUT GEOM. NONLINEARITY
C I
C I      SSHL : FOR 3D STRAIN FIELDS WITHOUT GEOM.  NONLINEARITY
C I             IN SHELLS
C I
C =================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C JACB3D   B3DLS    MATMOD   EQUILB   SHNORM    DIRVEC
C GETTHK   JACSHL   TSHR     BSHL     BTSHL
C
C***********************************************************
      REAL*8 N,NXI,NETA,NSI,NX,NY,NZ,NXASI,NYASI,NZASI,NSIX,NSIY,NSIZ
      REAL*8 LTHICK
      REAL*4 SHELLZ,THICK,XYZ
      INTEGER ELNUM
      COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
      COMMON/TRANS/V11(3),V12(3),V13(3)
```

```
      COMMON/SHLDIR/VECT(3,3,9)
      COMMON/INPUTE/ISPB(10000)
      COMMON/SKTR2/SHELLZ(3,10000)
      COMMON/ISHAP2/W(27)
      COMMON/ISHAP1/N(20,27),NXI(20,27),NETA(20,27),NSI(20,27),SI(27)
      COMMON/INPUT3/XYZ(3,10000)
      COMMON/INPUT2/NOP(20,5000)
      COMMON/MAIN2/UTOTAL(60000)
      COMMON/ASSEM2/II(120)
      COMMON/JACOB1/NX(20),NY(20),NZ(20),SIX,SIY,SIZ
      COMMON/B2/DUDX,DVDX,DWDX,DUDY,DVDY,DWDY,DUDZ,DVDZ,DWDZ,A5
      COMMON/INPUT9/THICK(9),IFLAG
      COMMON/ELSTR1/STRN(6)
      DIMENSION DOF(54),V1(3),V2(3)
C
C
C
      ENTRY S3DLS (ELNUM,ITYPE,NNEL,NRB,NCB,NIP,MATNUM,NNDF,NENDF,IOUT)
C
      DO 110 INTGPN = 1, NIP
         CALL JACB3D (INTGPN, ELNUM, NNEL, IERROR, DETJAC)
         CALL B3DLS (NNEL)
         CST = DETJAC*W(INTGPN)
C
         DUDX = 0.
         DVDX = 0.
         DWDX = 0.
         DUDY = 0.
         DVDY = 0.
         DWDY = 0.
         DUDZ = 0.
         DVDZ = 0.
         DWDZ = 0.
C
C*VDIR: ASSUME COUNT(20)
         DO 100 K1 = 1, NNEL
            K11 = NNDF*(K1-1) + 1
            K12 = K11 + 1
            K13 = K12 + 1
            DUDX = DUDX + NX(K1)*UTOTAL(II(K11))
            DVDX = DVDX + NX(K1)*UTOTAL(II(K12))
            DWDX = DWDX + NX(K1)*UTOTAL(II(K13))
            DUDY = DUDY + NY(K1)*UTOTAL(II(K11))
            DVDY = DVDY + NY(K1)*UTOTAL(II(K12))
            DWDY = DWDY + NY(K1)*UTOTAL(II(K13))
            DUDZ = DUDZ + NZ(K1)*UTOTAL(II(K11))
            DVDZ = DVDZ + NZ(K1)*UTOTAL(II(K12))
            DWDZ = DWDZ + NZ(K1)*UTOTAL(II(K13))
  100    CONTINUE
C
         STRN(1) = DUDX
         STRN(2) = DVDY
         STRN(3) = DWDZ
         STRN(4) = DUDY + DVDX
         STRN(5) = DWDY + DVDZ
         STRN(6) = DWDX + DUDZ
C
         CALL MATMOD (ELNUM, ITYPE, MATNUM, INTGPN, IFLAG, IOUT, DETJAC
     1        , 1, NNEL)
         CALL EQUILB (CST, NCB, NRB, NNDF, NENDF)
  110 CONTINUE
C
      RETURN
```

```
C
      ENTRY SSHL (ELNUM, ITYPE, NNEL, NRB, NCB, NIP, MATNUM, NNDF, NENDF
     1    , IDIM, IOUT)
C
C ----- evaluate the local shell coordinates of the nodes
C
      DO 120 K1 = 1, NNEL
          KP = NOP(K1,ELNUM)
          ICODE = IAND(ISPB(KP),4)
C
C ---- If shell rotations are assebled in the local shell coordinate
C       system then retreive the local Z-axis from storage. Else
C       evaluate the normal to the shell mid plane by a call to the
C       SHNORM routine.
C
          IF (ICODE .GT. 0) THEN
              CALL SHNORM (ELNUM, K1, NNEL, ITYPE, VECT(1,1,K1),VECT(1,2
     1            ,K1),VECT(1,3,K1))
          ELSE
              VECT(1,3,K1) = DBLE(SHELLZ(1,KP))
              VECT(2,3,K1) = DBLE(SHELLZ(2,KP))
              VECT(3,3,K1) = DBLE(SHELLZ(3,KP))
              CALL DIRVEC (VECT(1,1,K1),VECT(1,2,K1),VECT(1,3,K1))
          ENDIF
  120 CONTINUE
C
      NDIF = NNDF - NENDF
      DO 170 K1 = 1, NNEL
          NODE = NOP(K1,ELNUM)
          ID = NENDF*(K1-1)
          DO 130 K2 = 1, IDIM
              ID1 = ID + K2
              DOF(ID1) = UTOTAL(II(ID1))
  130     CONTINUE
          ID = ID + IDIM
          ICODE = IAND(ISPB(NODE),4)
          IF (ICODE .GT. 0) THEN
              DO 150 K2 = 1, IDIM
                  ID1 = ID + K2
                  TEMP = 0.
                  DO 140 K3 = 1, IDIM
                      ID2 = ID + K3
                      TEMP = TEMP + VECT(K3,K2,K1)*UTOTAL(II(ID2))
  140             CONTINUE
                  DOF(ID1) = TEMP
  150         CONTINUE
          ELSE
              DO 160 K2 = 1, IDIM
                  ID1 = ID + K2
                  DOF(ID1) = UTOTAL(II(ID1))
  160         CONTINUE
          ENDIF
  170 CONTINUE
C
      DO 190 INTGPN = 1, NIP
          CALL GETTHK (INTGPN, ELNUM, NNEL, THICKE, RAD)
C
C ----- vector V13 returned by JACSHL is normal to mid surface of the
C       shell at integration points
C
          ISET = 0
          SIP = SI(INTGPN)
          CALL JACSHL(INTGPN,ELNUM,NNEL,THICKE,DETJAC,V13,ISET,SIP)
```

```
C
C ----- the coordinates vectors V11 and V12 are evaluated by DIRVEC
C       which is part of the Element Library Module
C
          CALL DIRVEC (V11, V12, V13)
          IF(NLAYRS.GT.1)CALL TSHR(ELNUM,NNEL,INTGPN,THICKE,ITYPE)
          CALL BSHL (ELNUM, NNEL, INTGPN, THICKE)
          CST = DETJAC*W(INTGPN)
C
          DUDX = 0.
          DVDX = 0.
          DWDX = 0.
          DUDY = 0.
          DVDY = 0.
          DWDY = 0.
          DUDZ = 0.
          DVDZ = 0.
          DWDZ = 0.
C
C*VDIR: ASSUME COUNT(8)
C
          ASI = SI(INTGPN)
          CONST = THICKE*0.5D0
          DO 180 K1 = 1, NNEL
C
C ----- get the local shell coordinates of the node
C
              V1(1) = VECT(1,1,K1)
              V2(1) = VECT(1,2,K1)
              V1(2) = VECT(2,1,K1)
              V2(2) = VECT(2,2,K1)
              V1(3) = VECT(3,1,K1)
              V2(3) = VECT(3,2,K1)
C
              K11 = NNDF*(K1-1) + 1
              K12 = K11 + 1
              K13 = K12 + 1
              K14 = K13 + 1
              K15 = K14 + 1
              NSIX = N(K1,INTGPN)*SIX
              NSIY = N(K1,INTGPN)*SIY
              NSIZ = N(K1,INTGPN)*SIZ
              NXASI = NX(K1)*ASI
              NYASI = NY(K1)*ASI
              NZASI = NZ(K1)*ASI
C
              DUDX = DUDX + NX(K1)*DOF(K11) + CONST*((-V2(1)*DOF(K14))+
     1            V1(1)*DOF(K15))*(NXASI+NSIX)
              DVDX = DVDX + NX(K1)*DOF(K12) + CONST*((-V2(2)*DOF(K14))+
     1            V1(2)*DOF(K15))*(NXASI+NSIX)
              DWDX = DWDX + NX(K1)*DOF(K13) + CONST*((-V2(3)*DOF(K14))+
     1            V1(3)*DOF(K15))*(NXASI+NSIX)
              DUDY = DUDY + NY(K1)*DOF(K11) + CONST*((-V2(1)*DOF(K14))+
     1            V1(1)*DOF(K15))*(NYASI+NSIY)
              DVDY = DVDY + NY(K1)*DOF(K12) + CONST*((-V2(2)*DOF(K14))+
     1            V1(2)*DOF(K15))*(NYASI+NSIY)
              DWDY = DWDY + NY(K1)*DOF(K13) + CONST*((-V2(3)*DOF(K14))+
     1            V1(3)*DOF(K15))*(NYASI+NSIY)
              DUDZ = DUDZ + NZ(K1)*DOF(K11) + CONST*((-V2(1)*DOF(K14))+
     1            V1(1)*DOF(K15))*(NZASI+NSIZ)
              DVDZ = DVDZ + NZ(K1)*DOF(K12) + CONST*((-V2(2)*DOF(K14))+
     1            V1(2)*DOF(K15))*(NZASI+NSIZ)
              DWDZ = DWDZ + NZ(K1)*DOF(K13) + CONST*((-V2(3)*DOF(K14))+
```

```
    1            V1(3)*DOF(K15))*(NZASI+NSIZ)
  180     CONTINUE
C
          STRN(1) = DUDX
          STRN(2) = DVDY
          STRN(3) = DWDZ
          STRN(4) = DUDY + DVDX
          STRN(5) = DWDY + DVDZ
          STRN(6) = DWDX + DUDZ
C
          CALL MATMOD (ELNUM, ITYPE, MATNUM, INTGPN, IFLAG, IOUT, DETJAC
    1          , 1, NNEL)
          CALL BTSHL (ELNUM, NNEL, 6, NRB, 3, 2)
          CALL EQUILB (CST, NCB, NRB, NNDF, NENDF)
  190 CONTINUE
C
      RETURN
      END
C
C ======================== E Q U L I B ==============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE EQUILB(CST,NCB,NRB,NNDF,NENDF)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C**************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C**************************************************************
      REAL*8 B,RE,STRS,CST,TEMP
      COMMON/ASSEM2/II(120)
      COMMON/ELSTR2/STRS(6)
      COMMON/MAIN4/RE(60000)
      COMMON/B1/B(6,60)
C     DIMENSION RTEMP( 60 )
C
C*VDIR: ASSUME COUNT(16)
C*VDIR: IGNORE RECRDEPS
C
C     DO 200 K1 = 1 , NCB
C     TEMP = 0.
C     DO 100 K2 = 1 , NRB
C 100 TEMP = TEMP + B(K2 , K1)*STRS( K2 )
C 200 RTEMP( K1 ) =  TEMP*CST
C
      NDIF = NNDF - NENDF
      DO 110 K1 = 1, NCB
          ID1 = (K1-1)/NENDF
          ID2 = K1 + ID1*NDIF
          TEMP = 0.
          DO 100 K2 = 1, NRB
              TEMP = TEMP + B(K2,K1)*STRS(K2)
  100     CONTINUE
          RE(II(ID2)) = RE(II(ID2)) + TEMP*CST
  110 CONTINUE
C
      RETURN
      END
C
```

```
C ========================= G L O B A L ============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE GLOBAL
C
C ==================================================================
C I
C I     SUBROUTINE GLOBAL IS USED TO MODIFY THE FINAL GLOBAL
C I     STIFFNESS MATRIX. THIS IS DONE IN ORDER TO SOLVE THE
C I     SET OF SIMULTINOUS EQUATIONS BY THE METHOD OF MODIFFICATION.
C I     THIS SUBROUTINE IS DESIGED FOR MODIFICATION OF BANDED
C I     NONSYMETRIC MATRICES IN THEIR CONDENSED FORM.
C I
C I
C ==================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C************************************************************
      COMMON/MPCS/COEFMP(40000),ALAMB(40000),MPCDOF(40000),
     $           MPCADR(2,5000),NMPC,MPCPNT,MAXMPC
      DIMENSION IDOF( 1 )
C
C
C                  E N T R Y     G L O B 1
C
      ENTRY GLOB1(NNODES,NNDF,NTDF,IDOF)
C
      MDOF = NNDF*NNODES
      ICOUNT = 0
C
C*VDIR: PREFER SCALAR
      DO 100 ID = 1, MDOF + NMPC
          IF (IDOF(ID) .EQ. 0) THEN
              ICOUNT = ICOUNT + 1
              IDOF(ID) = ICOUNT
          ELSE IF (IDOF(ID) .GT. 0) THEN
              IDOF(ID) = 0
          ENDIF
  100 CONTINUE
C
      NTDF = ICOUNT
C
      RETURN
C
C
C                  E N T R Y     G L O B 2
C
      ENTRY GLOB2 (NNODES, NNDF, NTDF, IDOF)
C
      MDOF = NNDF*NNODES
      ICOUNT = 0
C
C*VDIR: PREFER SCALAR
      DO 110 ID = 1, MDOF + NMPC
          IF (IDOF(ID) .GT. 0) THEN
```

```
            ICOUNT = ICOUNT + 1
            IDOF(ID) = ICOUNT
         ENDIF
  110 CONTINUE
      NTDF = ICOUNT
C
      RETURN
      END
C ====================== C O O R D ======================
C
C     INCLUDE (PROCESS)
      SUBROUTINE COORD
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C SHNORM    DIRVEC
C
C*************************************************************
      REAL*4 XYZ,SHELLZ,THICK
      REAL*8 N,NXI,NETA,NSI,SI
      INTEGER ELNUM

      COMMON/MAIN2/UTOTAL(60000)
      COMMON/INPUT3/XYZ(3,10000)
      COMMON/INPUT2/NOP(20,5000)
      COMMON/ISHAP1/N(20,27),NXI(20,27),NETA(20,27),NSI(20,27),SI(27)
      COMMON/INPUT9/THICK(9),IFLAG
      COMMON/SKTR2/SHELLZ(3,10000)
      COMMON/INPUTE/ISPB(10000)
      DIMENSION U( 3 ),V3(3),V1(3),V2(3)
C
C              E N T R Y    C O O R D 1
C
      ENTRY COORD1(ELNUM,NNEL,INTGPN,ITYPE,X1,Y1,Z1)
      X1 = 0.
      Y1 = 0.
      Z1 = 0.
C*VDIR: ASSUME COUNT(8)
      DO 100 K = 1, NNEL
         IF (IFLAG .EQ. 0) THEN
            X1 = X1 + N(K,INTGPN)*XYZ(1,NOP(K,ELNUM))
            Y1 = Y1 + N(K,INTGPN)*XYZ(2,NOP(K,ELNUM))
            Z1 = Z1 + N(K,INTGPN)*XYZ(3,NOP(K,ELNUM))
         ELSE IF (IFLAG .EQ. 4) THEN
            KP = NOP(K,ELNUM)
            ICODE = IAND(ISPB(KP),4)
C
C ---- If shell rotations are assembled in the local shell coordinate
C       system then retreive the local Z-axis from storage. Else
C       evaluate the normal to the shell mid plane by a call to the
C       SHNORM routine.
C
            CONST = 0.5*SI(INTGPN)
            IF (ICODE .GT. 0) THEN
               CALL SHNORM (ELNUM, K, NNEL, ITYPE, V1(1),V2(1),V3(1))
            ELSE
               V3(1) = DBLE(SHELLZ(1,KP))
               V3(2) = DBLE(SHELLZ(2,KP))
               V3(3) = DBLE(SHELLZ(3,KP))
```

```
               CALL DIRVEC (V1(1),V2(1),V3(1))
            ENDIF
            X1 = X1 + N(K,INTGPN)*(XYZ(1,NOP(K,ELNUM))+THICK(K)*CONST*
     1            V3(1))
            Y1 = Y1 + N(K,INTGPN)*(XYZ(2,NOP(K,ELNUM))+THICK(K)*CONST*
     1            V3(2))
            Z1 = Z1 + N(K,INTGPN)*(XYZ(3,NOP(K,ELNUM))+THICK(K)*CONST*
     1            V3(3))
         ENDIF
  100 CONTINUE
C
      RETURN
C
C              E N T R Y    C O O R D 2
C
      ENTRY COORD2 (ELNUM, NNEL, INTGPN, NNDF, UXIP, UYIP, UZIP)
      U(1) = O.
      U(2) = O.
      U(3) = O.
C*VDIR: PREFER SCALAR
      DO 120 K = 1, NNEL
C*VDIR: PREFER SCALAR
         DO 110 ID = 1, NNDF
            K1 = NNDF*(NOP(K,ELNUM)-1) + ID
            U(ID) = U(ID) + N(K,INTGPN)*UTOTAL(K1)
  110    CONTINUE
  120 CONTINUE
      UXIP = U(1)
      UYIP = U(2)
      UZIP = U(3)
      RETURN
      END
C

C ====================== M A T M O D =============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE MATMOD(ELNUM,ITYPE,MATNUM,INTGPN,IFLAG,IOUT,DETJAC,
     #                  ICODE,nnel)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C DCONST    ERRORS
C
C***********************************************************
      INTEGER ELNUM
      CHARACTER COMM*6,BUFFER*80,BUFF*80
      COMMON/INPUTF/MATYPE(10)
      COMMON/COMP2/COMM,BUFFER,BUFF
C
      I = MATYPE(MATNUM)
      IF (I .EQ. 1) THEN
         CALL DCONST (ELNUM, ITYPE, MATNUM, INTGPN, IFLAG, IOUT, DETJAC
     1        , ICODE, NNEL)
      ELSE
         WRITE (BUFFER, *) 'MATERIAL', MATNUM
         CALL ERRORS (18, 1, 'MATMOD')
         STOP
      ENDIF
```

```
C
      RETURN
      END
C=========================G E T T H I C K=======================
C     INCLUDE(PROCESS)
      SUBROUTINE GETTHK(INTGPN,ELNUM,NNEL,THICKE,RAD)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C************************************************************
      INTEGER ELNUM
      REAL*4 THICK,XYZ
      REAL*8 N,NXI,NETA,NSI,SI
      COMMON/INPUT3/XYZ(3,10000)
      COMMON/INPUT2/NOP(20,5000)
      COMMON/ISHAP1/N(20,27),NXI(20,27),NETA(20,27),NSI(20,27),SI(27)
      COMMON/INPUT9/ THICK(9),IFLAG
      IF (IFLAG .EQ. 3) THEN
          RAD = 0.0
          DO 100 K1 = 1, NNEL
              RAD = RAD + N(K1,INTGPN)*XYZ(1,NOP(K1,ELNUM))
  100     CONTINUE
          THICKE = 2.0*3.141562*RAD
      ELSE
          THICKE = 0.0
          RAD = 0.0
          DO 110 K1 = 1, NNEL
              THICKE = THICKE + N(K1,INTGPN)*THICK(K1)
  110     CONTINUE
      ENDIF
      RETURN
C CHANGES 7/18
      END
C
C ====================== D I A G N L ============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE DIAGNL(NELEM,NNDF,NTDF,IDOF,JDIAG,NTSK,NNODES,IFLAG2,
     1 IOUT)
C ================================================================
C I
C I    THIS PROGRAM COMPUTES THE VECTOR CONTAINING THE ADDRESSES
C I    OF THE DIAGNAL ELEMENTS OF THE STIFFNESS MATIX. IT ALSO
C I    CALCULATES THE BANDWIDTH AND THE AVERAGE BANDWIDTH OF THE
C I    OF THE STIFFNESS MATRIX AND PRINTS THESE STATISTICS.
C I
C I    A R G U M E N T     L I S T
C I
C I    NELEM     = TOTAL NUMBER OF ELEMENTS
C I
C I    NNDF      = NUMBER OF NODAL DEGREES OF FREEDOM
C I
C I    NTDF      = NUMBER OF TOTAL DEGREES OF FREEDOM
C I
C I    IDOF(I)   = VECTOR CONTAINING THE D.O.F. NUMBERS OF THE NODES
C I
C I    JDIAG(I)  = VECTOR CONTAINING THE ADDRESS OF THE DIAGNAL TERMS
```

```
C I                IN THE GLOBAL STIFFNESS MATRIX 'SKG'
C I
C I    NTSK      = NUMBER OF TERMS IN THE GLOBAL STIFFNESS MATRIX
C I
C I    MBAND     = HALF BANDWIDTH OF THE STIFFNESS MATRIX
C I
C I    IOUT      = OUTPUT DEVICE NUMBER
C I
C I
C I    C O M M O N    B L O C K S
C I
C I    NOP(I,J)  = MEMBER INCIDENCES
C I
C I
C I
C ================================================================
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ELINFO
C
C************************************************************
      INTEGER ELNUM
      COMMON/INPUT2/NOP(20,5000)
      COMMON/MPCS/COEFMP(40000),ALAMB(40000),MPCDOF(40000),
     $            MPCADR(2,5000),NMPC,MPCPNT,MAXMPC
      COMMON/ASSEM2/II(120)
      DIMENSION IDOF( 1 ),JDIAG( 1 )
C
      MDOF = NNDF*NNODES
      NTSK = 0
C*VDIR: PREFER VECTOR
      DO 100 K = 1, MDOF + NMPC
          JDIAG(K) = 10000000
  100 CONTINUE
C
      DO 150 ELNUM = 1, NELEM
          CALL ELINFO (ELNUM, ITYPE, NNEL, IFLAG, ISTART, LINES)
          IF (ITYPE .LE. 0) GO TO 140
C
          MINDOF = 10000000
          DO 120 NODE = 1, NNEL
              ID = NNDF*(NOP(NODE,ELNUM)-1)
              ID1 = NNDF*(NODE-1)
              DO 110 IDIR = 1, NNDF
                  II(ID1+IDIR) = ID + IDIR
                  ID2 = IDOF(ID+IDIR)
                  IF (ID2 .GT. 0) MINDOF = MINO(MINDOF,ID2)
  110         CONTINUE
  120     CONTINUE
C
C        AT THIS POINT THE HIGHT OF EACH COLUMN IS STORED IN JDIAG
C
          DO 130 ID1 = 1, NNEL*NNDF
              ID2 = II(ID1)
              JDIAG(ID2) = MINO(JDIAG(ID2),MINDOF)
  130     CONTINUE
  140     CONTINUE
  150 CONTINUE
```

```
C
c    modify for mpc locate smallest dof number
C
      IF (NMPC .GT. 0) THEN
          DO 170 K1 = 1, NMPC
              I1 = MDOF + K1
              ILOC = MPCADR(1,K1)
              DO 160 K2 = ILOC, ILOC + MPCADR(2,K1) - 1
                  IF (MPCDOF(K2) .GT. 0) THEN
                      ID = IDOF(MPCDOF(K2))
                      IF (ID .GT. 0) JDIAG(I1) = MINO(JDIAG(I1),ID)
                  ENDIF
  160         CONTINUE
  170     CONTINUE
      ENDIF
      DO 180 K1 = 1, MDOF + NMPC
          ID = IDOF(K1)
          IF (ID .GT. 0) JDIAG(ID) = ID - JDIAG(K1) + 1
  180 CONTINUE
C         LOCATION OF EACH DIAGNAL TERM WILL NOW BE STORED IN JDIAG
C
      IF (IFLAG2 .EQ. 0) THEN
          MHT = 1
          ID = 0
          DO 190 K = 1, NTDF + 1
              ID = ID + MHT
              MHT = JDIAG(K)
              JDIAG(K) = ID
  190     CONTINUE
          NTSK = JDIAG(NTDF+1) - JDIAG(1)
      ELSE
          ID = 0
          DO 200 K = 1, NTDF
              ID = ID + JDIAG(K)
              JDIAG(K) = ID
  200     CONTINUE
          NTSK = 2*JDIAG(NTDF)
      ENDIF
C
C         NTSK = NUMBER OF TERMS IN THE GLOBAL STIFFNESS MATRIX "SKG"
C
C
      WRITE (IOUT, 900) MDOF, NTDF, NTSK
      RETURN
  900 FORMAT(//1X,'NUMBER OF dofs = ',I8/1X,'active dofs ',
     1 I8/1X,'SIZE OF THE STIFFNESS MATRIX','=',i8)
      END
C
C************************************************************
C     INCLUDE(PROCESS)
      SUBROUTINE GTLTK(INTGPN,ELNUM,NNEL,THICKL,ZSI,LTHICK,ZS)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C************************************************************
      INTEGER ELNUM
      REAL*8 N,NXI,NETA,NSI,SI,LTHICK
```

```
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/ISHAP1/N(20,27),NXI(20,27),NETA(20,27),NSI(20,27),SI(27)
      DIMENSION LTHICK(9),ZS(9)
C
      THICKL = 0.0
      ZSI = 0.0
      DO 100 K1 = 1, NNEL
          THICKL = THICKL + N(K1,INTGPN)*LTHICK(K1)
          IF (NIPSI .EQ. 1) THEN
              ZSI = ZSI + N(K1,INTGPN)*ZS(K1)
          ELSE IF (NIPSI .EQ. 2) THEN
              ICP = INTGPN/(NIPXI*NIPETA)
              IP = INTGPN - ICP*NIPXI*NIPETA
              IF (ICP.EQ.0 .OR. ICP.EQ.1 .AND. IP.EQ.0) THEN
                  ZSI = ZSI + N(K1,INTGPN)*(ZS(K1)-LTHICK(K1)*0.57735/
     1                2.0)
              ELSE IF (ICP.EQ.1 .OR. ICP.EQ.2 .AND. IP.EQ.0) THEN
                  ZSI = ZSI + N(K1,INTGPN)*(ZS(K1)+LTHICK(K1)*0.57735/
     1                2.0)
              ENDIF
          ELSE IF (NIPSI .EQ. 3) THEN
              ICP = INTGPN/(NIPXI*NIPETA)
              IP = INTGPN - ICP*NIPXI*NIPETA
              IF (ICP.EQ.0 .OR. ICP.EQ.1 .AND. IP.EQ.0) THEN
                  ZSI = ZSI + N(K1,INTGPN)*(ZS(K1)-LTHICK(K1)*0.7745966/
     1                2.0)
              ELSE IF (ICP.EQ.1 .OR. ICP.EQ.2 .AND. IP.EQ.0) THEN
                  ZSI = ZSI + N(K1,INTGPN)*ZS(K1)
              ELSE IF (ICP.EQ.2 .OR. ICP.EQ.3 .AND. IP.EQ.0) THEN
                  ZSI = ZSI + N(K1,INTGPN)*(ZS(K1)+LTHICK(K1)*0.7745966/
     1                2.0)
              ENDIF
          ENDIF
  100 CONTINUE
      RETURN
      END
C ========================= C O N T R L 1===========================
C
C     INCLUDE (PROCESS)
      SUBROUTINE CNTRL1(SKG,SKGL,R,IDOF,JDIAG,NTSK,NTDF,IOUT,mband)
C
C =================================================================
C I
C I    P R O G R A M:
C I
C I    SUBROUTINE 'CONTRL' CONTROLS THE INCREMENTAL LOADING AND THE
C I    NEWTON RAPHSON ITERATIVE PROCESS FOR THE TOTAL LAGRANGIAN
C I    GEOMETRIC AND MATERIAL NONLINEARITIES.
C I
C I    A R G U M E N T   L I S T:
C I
C I    SKG(I)       =  GLOBAL STIFFNESS MATRIX STORED AS A ONE
C I                    DIMENSIONAL ARRAY
C I    R(I)         =  LOAD VECTOR
C I    IDOF(I)      =  VECTOR CONTAINING THE D.O.F. NUMBERS OF JOINTS
C I    JDIAG(I)     =  LOCATION OF THE DIAGNAL TERMS OF EACH COLUMN
C I                    IN THE GLOBAL STIFFNESS MATRIX 'SKG'
C I    NTSK         =  TOTAL NUMBER OF TERMS IN THE 'SKG' MATRIX
C I    NTDF         =  NUMBER OF TOTAL D.O.F. IN THE PROBLEM
C I                    NOT INCLUDING THE CONSTRAINED BOUNDARIES
C I    IOUT         =  OUTPUT DEVICE
C I    MBAND        =  HALF BAND WIDTH OF THE STIFFNESS MATRIX
C I
```

```
C I
C I    C O M M O N   B L O C K S
C I
C I    REFFER TO THE COMMON BLOCK DISCRIPTIONS.
C I
C ==================================================================
C
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C RECOV     RESTAT    RCONST    DIAGNL    DIRCOS    ASSEMB
C REWIN     SOLVE2    SOLVE1    GETSTR    SWAP3     CHKRC
C IOCOPY    STORE     OUTPUT    ARCHIV
C
C***********************************************************
      LOGICAL YES,NO
      COMMON/TRANS/DC(3,3)
      COMMON/INPUTE/ISPB(10000)
      COMMON/MAIN1/U(60000),RE1(60000)
      COMMON/MAIN2/UTOTAL(60000)
      COMMON/MAIN4/RE(60000)
      COMMON/INPUT7/RIT(60000),RINC(60000),UINC(60000)
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC1,MNIT,IFLAG1,IFLAG2,IDIM,
     1              NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC(10),LDCONT,INCPTR
      COMMON/INPUTB/FAC,FACNEW,FACLOW,FACHIG,ENRG1,NDIVER,ISTOP
      COMMON/INPUTG/IFLAG3,IOINTR,IFPLOT
      COMMON/CONTR1/INCREM,NIT
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/IALGTM/IFLAG5
      COMMON/SAFE4/IRDOF(60000),IUDOF(60000)
      COMMON/PATH/HISTX(10,30),HISTY(10,30),IPATH(10)
      COMMON/TEMP5/TEMP1(60000),TEMP2(60000)
      COMMON/MPCS/COEFMP(40000),ALAMB(40000),MPCDOF(40000),
     $           MPCADR(2,5000),NMPC,MPCPNT,MAXMPC
c changes 7/18
      DIMENSION R(1),SKG(1),SKGL(1),IDOF(1),JDIAG(1),DUMMY(3)
C
      BTIME = 0


      BPREV = 0
      FRACST = 0
      ICHECK = 0
      YES = .TRUE.
      NO = .FALSE.
      IREP = NNDF/IDIM
C
C       MDF       = MAXIMUM DEGREES OF FREEDOM INCLUDING THE SUPPORTS
C
      MDF = NNODES*NNDF
      IF (IOINTR .EQ. 0) IOINTR = NLINC(INCPTR)
C
C       IF THIS RUN IS A RESTART THEN RESTORE THE LAST CONVERGED VALUES
C       OF THE EQUILIBRIUM LOAD VECTOR AND THE TOTAL DISPLACEMENT VECTOR
C
      IF (IFLAG3 .EQ. 1) THEN
          CALL RECOV (FRACST, MDF, ISTART, NTDF, IDOF)
          BPREV = FRACST
          CALL RESTAT
```

```
          CALL RCONST
          CALL DIAGNL (NELEM, NNDF, NTDF, IDOF, JDIAG, NTSK, NNODES,
     1         IFLAG2, IOUT)
          IFINAL = ISTART + NLINC(INCPTR)
          ISTART = ISTART + 1
          ISAVE = IFLAG1
      ELSE
          ISTART = 1
          IFINAL = NLINC(INCPTR)
C
C       FOR THE FIRST ITERATION OF THE FIRST INCREMENT USE THE
C       GEOMETRIC LINEARITY ROUTINS.
C
C       IFLAG1 = 0; GEOMETRIC LINARITY
C              = 1; GEOMETRIC NON-LINEARITY
C       ISAVE  = DUMMY VARIABLE USED TO STORE THE VALUE OF 'IFLAG1'
C
          ISAVE = IFLAG1
          IFLAG1 = 0
      ENDIF
C
C       CALCULATE THE PROPER LOAD OR DISPLACEMENT INCREMENT
C
C       UINC( K ) = APPLIED INCREMENT OF DISPLACEMENT
C       U( K )    = TOTAL APPLIED DISPLACEMENTS
C       R( K )    = TOTAL APPLIED LOADS
C       RINC( K ) = INCREMENT OF APPLIED LOADS
C       RE( K )   = EQUILIBRIUM LOAD VECTOR
C       NLINC     = NUMBER OF LOAD INCREMENTS
C
C
C
C       ICOUNT = ITERATION COUNT FOR THE RUN
C       IOCNT  = INCREMENT COUNT FROM THE THE START OR SINCE THE LAST
C                OUTPUT. WHEN 'IOCNT' IS EQUAL TO 'IOINTR' A COMPLETE
C                OUTPUT WILL BE GENERATED.
C
      DO IM = 1, MDF
          TEMP1(IM) = U(IM)
          TEMP2(IM) = R(IM)
      END DO
  110 CONTINUE
      IF (INCPTR .GT. 1) THEN
          ISTART = IFINAL + 1
          IFINAL = NLINC(INCPTR) + ISTART - 1
      ENDIF
      IOCNT = 0
      IPLCNT = 0
C
C                     S T A R T     O F
C             I N C R E M E N T     L O O P
C
      DO 340 INCREM = ISTART, IFINAL
          IOCNT = IOCNT + 1
          IPLCNT = IPLCNT + 1
          RFACT = 1.0D0
          RFSUM = 0.0D0
          FAC = FACLOW
          FACNEW = FACHIG
C
C       FAC      = CONVERGANCE FACTOR
C       FACLOW   = LOWEST ALLOWABLE CONVERGENCE FACTOR
C       FACHIG   = LARGEST ALLOWABLE CONVERGENCE FACTOR
```

```
C       FACNEW   = NEW CONVERGANCE FACTOR CALCULATED BY ROUTINE 'CHECK'
C       RFACT    = REDUCTION FACTOR FOR SUBINCREMENTATION
C       U( K )   = INCREMENT OF THE APPLIED DISPLACEMENTS USED
C                     FOR THE FIRST ITERATION
C       RIT( K ) = TOTAL APPLIED LOAD AT THE END OF THE INCREMENT
C
C*VDIR: PREFER VECTOR
          DO 160 K1 = 1, MDF
              ATIME = 0.0
              BTIME = 0.0
              IF (INCPTR .EQ. 1) THEN
                  ATIME = FRACT(INCPTR)
                  BTIME = ATIME*INCREM/IFINAL
                  IF (IFLAG3 .EQ. 1) THEN
                      ATIME = FRACT(INCPTR) - FRACST
                      BTIME = FRACST + ATIME*(INCREM-ISTART+1)/NLINC(
     1                    INCPTR)
                  ENDIF
              ELSE IF (INCPTR .GT. 1) THEN
                  ATIME = FRACT(INCPTR) - FRACT(INCPTR-1)
                  BTIME = FRACT(INCPTR-1) + ATIME*(INCREM-ISTART+1)/
     1                NLINC(INCPTR)
              ENDIF
              IRHFN = 0
              IUHFN = 0
              NUM1 = 0
              NUM2 = 0
              FRACT1 = 0.0
              FRACT2 = 0.0
              IF (IRDOF(K1) .NE. 0) IRHFN = IRDOF(K1)
              IF (IUDOF(K1) .NE. 0) IUHFN = IUDOF(K1)
              IF (IRHFN .NE. 0) NUM1 = IPATH(IRHFN)
              IF (IUHFN .NE. 0) NUM2 = IPATH(IUHFN)
              IF (NUM1 .GT. 1) THEN
                  DO IP1 = 1, NUM1 - 1
                      IF (BTIME.GE.HISTX(IRHFN,IP1) .AND. BTIME.LE.HISTX
     1                    (IRHFN,IP1+1)) THEN
                          FRACT1 = (BTIME*(HISTY(IRHFN,IP1)-HISTY(IRHFN,
     1                        IP1+1))+(HISTX(IRHFN,IP1)*HISTY(IRHFN,IP1+
     2                        1)-HISTY(IRHFN,IP1)*HISTX(IRHFN,IP1+1)))/(
     3                        HISTX(IRHFN,IP1)-HISTX(IRHFN,IP1+1))
                          GO TO 130
                      ENDIF
                  END DO
              ENDIF
  130         CONTINUE
              IF (NUM2 .GT. 1) THEN
                  DO IP2 = 1, NUM2 - 1
                      IF (BTIME.GE.HISTX(IUHFN,IP2) .AND. BTIME.LE.HISTX
     1                    (IUHFN,IP2+1)) THEN
                          FRACT2 = (BTIME*(HISTY(IUHFN,IP2)-HISTY(IUHFN,
     1                        IP2+1))+(HISTX(IUHFN,IP2)*HISTY(IUHFN,IP2+
     2                        1)-HISTY(IUHFN,IP2)*HISTX(IUHFN,IP2+1)))/(
     3                        HISTX(IUHFN,IP2)-HISTX(IUHFN,IP2+1))
                          GO TO 150
                      ENDIF
                  END DO
              ENDIF
  150         CONTINUE
              U(K1) = RFACT*TEMP1(K1)*FRACT2
              RIT(K1) = RFACT*TEMP2(K1)*FRACT1
              UINC(K1) = U(K1)
              RINC(K1) = RIT(K1)
```

```
  160     CONTINUE
C
C                    S T A R T     O F
C             I T E R A T I O N     L O O P
C
          DO 320 NIT = 1, MNIT
C
C        NIT =  ITERATION NUMBER
C        MNIT = MAXIMUM NUMBER OF ITERATIONS ALLOWED
C
              DO 220 K1 = 1, NNODES
                  I = NNDF*(K1-1)
                  DO 210 ILOOP = 1, IREP
                      ILOOP1 = ILOOP - 1
                      ICODE = IAND(ISPB(K1),ILOOP)
                      I = I + IDIM*ILOOP1
C*VDIR: PREFER SCALAR
                      DO 170 K2 = 1, IDIM
                          IDIR = I + K2
                          DUMMY(K2) = RIT(IDIR)
  170                 CONTINUE
                      IF (ICODE .GT. 0) THEN
                          CALL DIRCOS (ICODE, IDIM, K1)
C*VDIR: PREFER SCALAR
                          DO 190 K2 = 1, IDIM
                              CST = 0.
C*VDIR: PREFER SCALAR
                              DO 180 K3 = 1, IDIM
                                  IDIR = I + K3
                                  CST = CST + RIT(IDIR)*DC(K3,K2)
  180                         CONTINUE
                              DUMMY(K2) = CST
  190                     CONTINUE
                      ENDIF
C*VDIR: PREFER SCALAR
                      DO 200 K2 = 1, IDIM
                          IDIR = I + K2
                          ID = IDOF(IDIR)
                          IF (ID .GT. 0) R(ID) = DUMMY(K2)
  200                 CONTINUE
  210             CONTINUE
  220         CONTINUE
C
              IF (NIT .EQ. 1) THEN
                  LDEV = LDEV1
              ELSE
                  LDEV = LDEV2
              ENDIF
C
              CALL ASSEMB(SKG,SKGL,R,U,IDOF,JDIAG,NTSK,MBAND,IOUT)
C
              CALL REWIN
C
              IF (IFLAG2 .EQ. 0) THEN
                  CALL SOLVE2 (SKG, R, JDIAG, NTDF, 1, IOUT)
                  CALL SOLVE2 (SKG, R, JDIAG, NTDF, 2, IOUT)
              ELSE IF (IFLAG2 .EQ. 1) THEN
                  CALL SOLVE1 (SKG, SKGL, R, JDIAG, NTDF, YES, YES)
              ENDIF
C
              IF (NMPC .GT. 0) THEN
                  DO IMPC = 1, NMPC
                      ALAMB(IMPC) = R((NTDF-NMPC)+IMPC)
```

```
                END DO
            ENDIF
C
C*VDIR: PREFER SCALAR
C
            DO 240 K1 = 1, MDF
                ID = IDOF(K1)
                IF (ID .GT. 0) U(K1) = U(K1) + R(ID)
  240       CONTINUE
C
            DO 300 K1 = 1, NNODES
                I = NNDF*(K1-1)
                DO 290 ILOOP = 1, IREP
                    ILOOP1 = ILOOP - 1
                    ICODE = IAND(ISPB(K1),ILOOP)
                    I = I + IDIM*ILOOP1
C*VDIR: PREFER SCALAR
                    DO 250 K2 = 1, IDIM
                        IDIR = I + K2
                        DUMMY(K2) = U(IDIR)
  250               CONTINUE
                    IF (ICODE .GT. 0) THEN
                        CALL DIRCOS (ICODE, IDIM, K1)
C*VDIR: PREFER SCALAR
                        DO 270 K2 = 1, IDIM
                            CST = 0.
C*VDIR: PREFER SCALAR
                            DO 260 K3 = 1, IDIM
                                IDIR = I + K3
                                CST = CST + DC(K2,K3)*U(IDIR)
  260                       CONTINUE
                            DUMMY(K2) = CST
  270                   CONTINUE
                    ENDIF
C*VDIR: PREFER SCALAR
                    DO 280 K2 = 1, IDIM
                        IDIR = I + K2
                        UTOTAL(IDIR) = DUMMY(K2)
  280               CONTINUE
  290           CONTINUE
  300       CONTINUE
c 550 U( IDIR ) = DUMMY( K2 )
C
            IFLAG1 = ISAVE
C*VDIR: PREFER VECTOR
            DO 310 K1 = 1, MDF
                RE1(K1) = RE(K1)
                U(K1) = 0.
                U(K1) = UINC(K1)
                RE(K1) = 0.
  310       CONTINUE
C
            CALL GETSTR (IOUT)
            CALL REWIN
C
C
C
C     check the convergence of the procedure for each element of
C     reinforced concrete.
C
            ICHECK = 0
            CALL SWAP3
            CALL CHKRC (INCREM, NIT, IERROR, IOUT, ICHECK)
```

```
            IF (ICHECK .EQ. 0) THEN
                CALL IOCOPY
                CALL REWIN
                GO TO 330
            ENDIF
  320   CONTINUE
C
C                      E N D      O F
C            I T E R A T I O N     L O O P
C
C
  330   CONTINUE
        IF (ICHECK.EQ.0 .OR. NIT.GE.MNIT) THEN
            WRITE (IOUT, 910) INCREM
            IF (NIT.GT.MNIT .AND. ICHECK.EQ.1) THEN
                CALL REWIN
                WRITE (IOUT, 900)
                WRITE (IOUT, 920) MNIT
                CALL STORE (BPREV, MDF, INCREM - 1, NTDF, IDOF)
                CALL OUTPUT (SKG, IOUT, IERROR)
                CALL REWIN
                CALL ARCHIV (MDF)
                STOP
            ENDIF
            BPREV = BTIME
            CALL STORE (BTIME, MDF, INCREM, NTDF, IDOF)
            WRITE (IOUT, 920) NIT
            CALL OUTPUT (SKG, IOUT, IERROR)
            CALL REWIN
        ENDIF
  340 CONTINUE
      INCPTR = INCPTR + 1
      IOINTR = 0
      IFLAG3 = 0
      IF (INCPTR .LE. LDCONT) GO TO 110
      CALL ARCHIV (MDF)
      RETURN
C


C
C
  900 FORMAT(35X,'     NO CONVERGENCE  ')
  910 FORMAT(35X,'    INCREMENT NO.',I7,)
  920 FORMAT(35X,' ITTERATION NO.',I7,'  ')
C1002 FORMAT(///1X,'------ TOTAL NUMBER OF ITERATIONS FOR THIS RUN IS'
C    1 ,' = ',I5)
  930 FORMAT(//1X,
     1 '------ OUTPUTS ARE FOR THE LAST CONVERGED INCREMENT ',I4)
  940 FORMAT(///////1X,'------ OUTPUTS AT INCREMENT ',I4)
  950 FORMAT(/1X,'------ ALLOWABLE NUMBER OF ITERATIONS EXCEEDED AT',
     1 ' LOAD STEP ',I4/9X,'THE EFFECTIVE CONVERGENCE FACTOR ',E10.3,
     2 ' IS WITHIN TOLERANCE '/9X,'EXECUTION CONTINUES')
  960 FORMAT(/1X,'------ ALLOWABLE NUMBER OF DIVERGING ITERATIONS',
     1 ' IS EXCEEDED AT LOAD STEP ',I4/
     2 9X,'LOAD STEP FACTOR IS REDUCED TO ',F6.4/
     3 9X,'EXECUTION CONTINUES')
  970 FORMAT(/1X,'------ ALLOWABLE NUMBER OF ITERATIONS EXCEEDED ',
     1 ' AT LOAD STEP ',I4/9X,'THE EFFECTIVE CONVERGENCE FACTOR',E10.3,
     2 ' EXCEEDS THE PRISCRIBED TOLERANCE'/
     3 9X,'SUBINCREMENTATION HAS FAILED TO CORRECT THE PROBLEM'/
     4 9X,'EXECUTION TERMINATED')
  980 FORMAT(/1X,'------ ALLOWABLE NUMBER OF DIVERGING ITERATIONS',
     1 ' EXCEEDED AT LOAD STEP ',I4/
```

```
     2 9X,'SUBINCREMENTATION HAS FAILED TO CORRECT THE PROBLEM'/
     3 9X,'EXECUTION TERMINATED')
  990 FORMAT(/1X,'------ ALLOWABLE NUMBER OF ITERATIONS EXCEEDED AT',
     1 ' LOAD STEP ',I4/9X,'THE EFFECTIVE CONVERGENCE FACTOR',E10.3,
     2 ' EXCEEDS THE PRISCRIBED TOLERANCE'/9X,'REDUCTION FACTOR IS ',
     3 'REDUCES TO ',F6.4/9X,'EXECUTION CONTINUES')
      END
c#######################################################################
C     INCLUDE(PROCESS)
      SUBROUTINE TSHR(ELNUM,NNEL,INTGPN,THICKE,ITYPE)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C GTLTK     DMATCS
C
C***************************************************************
      INTEGER ELNUM
      REAL*8 LTHICK
      REAL*4 THICK
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON/INPUT9/THICK(9),IFLAG
      COMMON/TSHR1/E1(5000,9),F1(5000,9)
      COMMON/MATER1/DEP(6,6)
      DIMENSION A(9),B(9),C(9),D(9)
c
      II = NIPXI*NIPETA
      IPOINT = INTGPN/II
      INDEX = INTGPN - IPOINT*II
      IF (INDEX .EQ. 0) INDEX = II
C
      IF (LYNUM.EQ.1 .AND. INTGPN.LE.II) THEN
          A(INDEX) = 0.0
          B(INDEX) = 0.0
          C(INDEX) = 0.0
          D(INDEX) = 0.0
          E1(ELNUM,INDEX) = 0.0
          F1(ELNUM,INDEX) = 0.0
      ENDIF
C
      ZSI = 0.0
      THICKL = 0.0
      CALL GTLTK (INTGPN, ELNUM, NNEL, THICKL, ZSI, LTHICK, ZS)
      IPG = 1
      IOUT = 13
      ICODE = 0
      CALL DMATCS (ELNUM, ITYPE, INTGPN, IFLAG, IOUT, ICODE, IPG)
C
      ZDIST = THICKE/2.0 + ZSI
      A(INDEX) = THICKL*ZDIST*DEP(1,1) + A(INDEX)
      B(INDEX) = THICKL*DEP(1,1) + B(INDEX)
      C(INDEX) = THICKL*ZDIST*DEP(2,2) + C(INDEX)
      D(INDEX) = THICKL*DEP(2,2) + D(INDEX)
C
      IF (LYNUM .EQ. NLAYRS) THEN
          ISIG = 1
      ELSE
```

```
          ISIG = 0
       ENDIF
C
       IF (ISIG .EQ. 1) THEN
          E1(ELNUM,INDEX) = A(INDEX)/B(INDEX)
          F1(ELNUM,INDEX) = C(INDEX)/D(INDEX)
       ENDIF
       RETURN
       END
C
C =================== B L O C K   D A T A ========================
C
C      INCLUDE (PROCESS)
       BLOCK DATA
       IMPLICIT REAL*8 (A-H,O-Z)
       REAL*4 XII,ETAI,SII,FMAG,DMAG,PXI,PETA
       CHARACTER*80 GTITLE
       CHARACTER*40 SCFP,OFP,OFN,VOLSER
C
C      CONSTANT VALUES ASSIGNED TO CERTAIN VARIABLE NAMES.
C
       COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC1,MNIT,IFLAG1,IFLAG2,IDIM,
      1              NINODE,NCOLOR,NFREE
       COMMON/INCR11/FRACT(10),NLINC(10),LDCONT,INCPTR
       COMMON/INPUTB/FAC,FACNEW,FACLOW,FACHIG,ENRG1,NDIVER,ISTOP
       COMMON/INPUTG/IFLAG3,IOINTR,IFPLOT
       COMMON/UTIL3/NREC(3),NWMAX
       COMMON/FILE1/KEEP,IDEL,SCFP,OFP,OFN,VOLSER
       COMMON/ADMAT1/AD(81)
       COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
       COMMON/ELLIB1/XII(20),ETAI(20),SII(20)
       COMMON/ELLIB2/PXI(9),PETA(9)
       COMMON/HERM/H(4 , 4),GX(4 , 6),GY(4 , 6)
       COMMON/GRAPH1/IS(62),IE(62)
       COMMON/GRAPH5/FMAG,DMAG,HIGHT,ANGLE,NOLINE,ITHICK,NLINES,NLIN
       COMMON/GRAPH7/GTITLE(20)
       COMMON/GRAPH8/LDEVP
       COMMON/EXTRP1/INT33(9),INT22(4)
       COMMON/CAEDS1/I204(4),I208(8),I308(8),I320(20)
C
       DATA SCFP,OFP,OFN,VOLSER/' ',' ',' ',' '/
       DATA KEEP,IDEL/0,0/
       DATA ((H(I,J),J=1,4),I=1,4)/2.,-2.,1.,1.,-3.,3.,-2.,-1.,0.,0.,
      1     1.,0.,1.,0.,0.,0./
       DATA LDEV5/16/,LDEVP/17/
       DATA LDEV1,LDEV2,LDEV3,LDEV4,LDEVST/1,2,3,4,14/
       DATA (XII(K),K=1,20)/-1.,1.,1.,-1.,-1.,1.,1.,-1.,0.,1.,0.,-1.,
      1   -1.,1.,1.,-1.,0.,1.,0.,-1./
       DATA (ETAI(K),K=1,20)/-1.,-1.,1.,1.,-1.,-1.,1.,1.,-1.,0.,1.,
      1    0.,-1.,-1.,1.,1.,-1.,0.,1.,0./
       DATA (NLINC(K),K=1,10)/1,1,1,1,1,1,1,1,1,1/
       DATA (FRACT(K),K=1,10)/1.,1.,1.,1.,1.,1.,1.,1.,1.,1./
       DATA (SII(K),K=1,20)/-1.,-1.,-1.,-1.,1.,1.,1.,1.,-1.,-1.,-1.,-1.,
      1    0.,0.,0.,0.,1.,1.,1.,1./
       DATA NNODES,NELEM,NNDF,MNIT,IFLAG1,IFLAG2,IDIM/0,0,2,1,0,
      1    0,2/,IFLAG3,IOINTR,IFPLOT,NFREE/0,0,0,0/
       DATA NDIVER,FAC/1,.001/,AD/81*0./
       DATA NREC/1,1,1/,NWMAX/0/
       DATA PXI/-1.,1.,1.,-1.,0.,1.,0.,-1.,0./
       DATA PETA/-1.,-1.,1.,1.,-1.,0.,1.,0.,0./
C
C ----- GRAPHICS ELEMENT LINE CONECTIVITY DATA
C
```

```
      DATA IS/1,2,3,4,1,5,2,3,4,
     1        1,5,2,6,3,7,4,8,1,2,3,4,5,6,7,8,2,3,1,4,
     2    1,9,2,10,3,11,4,12,5,17,6,18,7,19,8,20,1,13,4,16,3,15,2,14,
     3    1,5,6,2,7,3,8,4,9/
      DATA IE/2,3,4,1,5,2,3,4,1,
     1        5,2,6,3,7,4,8,1,2,3,4,1,6,7,8,5,6,7,5,8,
     2    9,2,10,3,11,4,12,1,17,6,18,7,19,8,20,5,13,5,16,8,15,7,14,6,
     3    5,6,2,7,3,8,4,9,1/
      DATA FMAG,DMAG,HIGHT,ANGLE,NOLINE,ITHICK/1.,1.,0.08,0.,0,2/
      DATA NLINES,NLIN/0,0/
C
C       GAUSSIAN POINT TO NODE CONNECTIVITY DATA FOR NODAL EXTRAPOLATION
C
      DATA INT33/1,3,9,7,2,6,8,4,5/,INT22/1,2,4,3/
C
C       ELEMENT CONNECTIVITY DATA FOR CAEDS ELEMENTS
C
      DATA I204/1,2,3,4/
      DATA I208/1,5,2,6,3,7,4,8/
      DATA I308/1,2,3,4,5,6,7,8/
      DATA I320/1,9,2,10,3,11,4,12,13,14,15,16,5,17,6,18,7,19,8,20/
      END
C
C ========================== C A E D S ============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE CAEDS(IDOF,U,R,IOUT,UNIVER)
C
C =================================================================
C I
C I   P R O G R A M:
C I
C I   CAEDS is used to read the Universal Files which have been
C I   created by the CAEDS Graphics Finite Element Modeler module.
C I
C I
C I   O N   E N T R Y:
C I
C I   IOUT   = output device number
C I
C I
C I   O N   R E T U R N:
C I
C I   IDOF   = is the array containing the degree of freedom
C I            information for each node.
C I            =1; degree of freedom is constrained (zero displ.)
C I            =0; degree of freedom is unconstrained
C I            =-1; degree of freedom has a prescribed value and
C I                 is constrained (non-zero displacement)
C I
C =================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ERRORS
C
C************************************************************
      REAL*8 NUX,NUY,NUZ
```

```
      REAL*4 XYZ,XAX,ORIG,XZPLN
      CHARACTER*80 BUFFER,ID1,ID2,ID3,ID4
      CHARACTER*40 UNIVER,UNV,RSNAME,csname
      COMMON/INPUT2/NOP(20,5000)
      COMMON/INPUT3/XYZ(3,10000)
      COMMON/INPUT5/NUX(10),NUY(10),NUZ(10),EX(10),EY(10),EZ(10),
     1 P1X(10),P1Y(10),P1Z(10),P2X(10),P2Y(10),P2Z(10)
      COMMON/INPUT6/WGTX(10),WGTY(10),WGTZ(10)
      COMMON/OUTPT1/IOFLAG,IFCAED
      COMMON/CAEDS2/ID1,ID2,ID3,ID4,LOADCN
C
C ---- Common INPUT7 is used with a different organization in module
C       SAFE. It is used here for the porpuse of saving storage for
C       holding some temperary parameters. Array dummy is used to
C       adjust the size of this common block to match the size specified
C       in module SAFE.
C
      COMMON/INPUT7/IDEF(10000),IDISP(10000),ORIG(3,10000),XAX(3,10000),
     1              XZPLN(3,10000),ICTYPE(10000),IREF(10000),
     2              NPHYS(10000),DUMMY(110000)
C
C ---- IDEF    = definition coordinate system number for the node
C       IDISP   = displacement coordinate system number
C       ORIG    = coordinates of the origin
C       XAX     = coordinates of a point on the xaxis
C       XZPLN   = coordinates of a point on the xz-plane
C       ICTYPE  = coordinate type
C       IREF    = reference coordinate system of the current coor. sys.
C       NPHYS   = physical property number of the element
C
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1              NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC1(10),LDCONT,INCPTR
      COMMON/INPUTA/INFOEL(6,5000)
      COMMON/INPUTE/ISPB(10000)
      COMMON/INPUTF/MATYPE(10)
      COMMON/FREEB/IFBODY(10000)
      COMMON/MPCS/coefmp(40000),ALAMB(40000),MPCDOF(40000),
     $           MPCADR(2,5000),nmpc,mpcpnt,maxmpc
      DIMENSION IDOF(NNDF,*),U(NNDF,*),R(NNDF,*),NVAR(8)
      DIMENSION IDN(8),ISW(6)
      EQUIVALENCE(RSNAME,CSNAME)
C
C ---- IFCAED is the flag that indicates to the output module
C       that the requsted outputs should also be provided in the
C       form of a CAEDS universal file.
C
      NUM = 0
      ITYPE = 0
      INTCOD = 0
      NIPSI = 0
      NIPETA = 0
      NIPXI = 0
      LINES = 0
      ISTART = 0
      ITYPE1 = 0
      IFLAG = 0
      IFCAED = 1
      ID2 = 'NLRC3D STATIC ANALYSIS'
C
C ---- Open the CAEDS universal file
C
      UNV = '/'//UNIVER
```

```
      OPEN(19, STATUS='UNKNOWN', FILE=UNV)
C
      NHIGH = 0
      NLOW = 100000
      LHIGH = 0
      LLOW = 100000
C
      IOFLAG = IBSET(IOFLAG,20)
      IOFLAG = IBSET(IOFLAG,22)
C
      IDSAT = 0
  100 CONTINUE
      ICOUNT = 0
      IF (IDSAT .EQ. 0) THEN
          READ (19, '(A80)', END=300) BUFFER
          IF (BUFFER(5:6) .EQ. ('-1')) IDSAT = 1
          GO TO 100
      ELSE IF (IDSAT .EQ. 1) THEN
          READ (19, *, END=310) ITYPE
C
          IF (ITYPE .EQ. 15) THEN
              GO TO 120
          ELSE IF (ITYPE .EQ. 18) THEN
              GO TO 130
          ELSE IF (ITYPE .EQ. 71) THEN
              GO TO 140
          ELSE IF (ITYPE .EQ. 749) THEN
              GO TO 290
          ELSE IF (ITYPE .EQ. 752) THEN
              GO TO 150
          ELSE IF (ITYPE .EQ. 754) THEN
              GO TO 180
          ELSE IF (ITYPE .EQ. 755) THEN
              GO TO 240
          ELSE IF (ITYPE .EQ. 756) THEN
              GO TO 270
          ELSE
              ICOUNT = 0
  110         CONTINUE
              READ (19, '(a80)', END=310) BUFFER
              ICOUNT = ICOUNT + 1
              IF (BUFFER(5:6) .NE. ('-1')) GO TO 110
C
              ICOUNT = ICOUNT - 1
              WRITE (IOUT, 900) ICOUNT, ITYPE
              IDSAT = 0
              GO TO 100
          ENDIF
      ENDIF
C
C
C ---- Data set 15:  NODES
C
C ---- Dataset Organization
C
C       Record 1:       FORMAT(4I10,1P3E13.5)
C                       Field 1     -- Node lable
C                       Field 2     -- Definition coord. system number
C                       Field 3     -- Displacement coord. system number
C                       Field 4     -- color
C                       Fields 5-7  -- 3-Dimensional coordinate of the
C                                      node in the definition system
C
```

```
C       Record 1 is repeated for each node in the model.
C
C
  120 CONTINUE
      READ (19, '(a80)', END=310) BUFFER
      IF (BUFFER(5:6) .NE. ('-1')) THEN
          ICOUNT = ICOUNT + 1
C
C ---- ISKIP = dummy variable to read the color of the node
C
          READ (BUFFER, '(4I10,1P,3E13.5)') NODE, IDEF(NODE), IDISP(NODE
     1        ), ISKIP, XYZ(1,NODE), XYZ(2,NODE), XYZ(3,NODE)
C
C ---- Set bit 10 of ISPB to 1 to indicate that the position of the
C       node has been defined.
C
          ISPB(NODE) = IBSET(ISPB(NODE),10)
          NHIGH = MAX0(NHIGH,NODE)
          NLOW = MIN0(NLOW,NODE)
          GO TO 120
      ELSE
          WRITE (IOUT, 910) ICOUNT, ITYPE, NHIGH, NLOW
          IDSAT = 0
          GO TO 100
      ENDIF
C
C
C ---- Data set 18:  Coordinate Systems
C
C
C ---- Dataset Organization
C
C       Record 1:       FORMAT(5I10)
C                       Field 1     -- Coordinate system number
C                       Field 2     -- Coordinate system type
C                                      =0 - cartesian
C                                      =1 - cylindrical
C                                      =2 - spherical
C                       Field 3     -- Reference coordinate system num.
C                       Field 4     -- Color
C                       Field 5     -- Method of definition
C                                      =1 - origin, +X axis, +XZ plane
C
C
C       Record 2:       FORMAT(20A2)


C                       Field 1     -- Coordinate system name
C
C
C       Record 3:       FORMAT(1P6E13.5)
C                       Total of nine coordinate system definition param.
C                       Field 1-3   -- Origin of new system in reference
C                                      system
C                       Field 4-6   -- Point on +X axis of the new system
C                                      specified in refernce system
C                       Field 5     -- Point on +XZ plane of new system
C                                      specified in reference system
C
C       Record 1 through 3 are repeated for each coordinate system in
C       the model.
C
  130 CONTINUE
      READ (19, '(a80)', END=310) BUFFER
```

```
      IF (BUFFER(5:6) .NE. ('-1')) THEN
          ICOUNT = ICOUNT + 1
          READ (BUFFER, '(5I10)') NUM, ICTYPE(NUM), IREF(NUM), ISKIP,
     1        METHOD
          READ (19, *) BUFFER
          READ (19, '(1P,6E13.5)') (ORIG(I,NUM), I = 1, 3), (XAX(I,NUM)
     1         , I = 1, 3)
          READ (19, '(1P,6E13.5)') (XZPLN(I,NUM), I = 1, 3)
          GO TO 130
      ELSE
          IDSAT = 0
          WRITE (IOUT, 920) ICOUNT, ITYPE
          GO TO 100
      ENDIF
C
C
C
C ---- Data set 71:  Elements
C
C                  Table of Element Names and Type Numbers
C
C       Elem_Type       N_Nodes     Description
C
C          11              2        Rod
C          21              2        Linear Beam
C          22              2        Tapered Beam
C          23              2        Curved Beam
C          24              3        Parabolic Beam
C          31              2        Straight Pipe
C          32              2        Curved Pipe
C          41              3        Plane stress Triangle
C          42              6        Plane stress parabolic triangle
C          43              9        Plane stress cubic triangle
C          44              4        Plane stress linear quadrilateral
C          45              8        Plane stress parabolic quadrilateral
C          46             12        Plane stress cubic quadrilateral
C          51              3        Plane strain Triangle
C          52              6        Plane strain parabolic triangle
C          53              9        Plane strain cubic triangle
C          54              4        Plane strain linear quadrilateral
C          55              8        Plane strain parabolic quadrilateral
C          56             12        Plane strain cubic quadrilateral
C          61              3        Flat plate linear triangle
C          62              6        Flat plate parabolic triangle
C          63              9        Flat plate cubic triangle
C          64              4        Flat plate linear quadrilateral
C          65              8        Flat Plate parabolic quadrilateral
C          66             12        Flat plate cubic quadrilateral
C          71              4        Membrain linear quadrilateral
C          72              6        Membrain parabolic triangle
C          73              9        Membrain cubic triangle
C          74              3        Membrain linear triangle
C          75              8        Membrain parabolic quadrilateral
C          76             12        Membrain cubic quadrilateral
C          81              3        Axisymmetric linear triangle
C          82              6        Axisymmetric parabolic triangle
C          84              4        Axisymmetric linear quadrilateral
C          85              8        Axisymmetric parabolic quadrilateral
C          91              3        Thin shell linear triangle
C          92              6        Thin shell parabolic triangle
C          93              9        Thin shell cubic triangle
C          94              4        Thin shell linear quadrilateral
C          95              8        Thin shell parabolic quadrilateral
```

```
C         96              12        Thin shell cubic quadrilateral
C        101              6         Thick shell linear wedge
C        102              12        Thick shell parabolic wedge
C        103              18        Thick shell cubic wedge
C        104              8         Thick shell linear brick
C        105              16        Thick shell parabolic brick
C        106              24        Thick shell cubic brick
C        111              4         Solid linear tetrahedron
C        112              6         Solid linear wedge
C        113              15        Solid parabolic wedge
C        114              24        Solid cubic wedge
C        115              8         Solid linear brick
C        116              20        Solid parabolic brick
C        117              32        Solid cubic brick
C        118              10        Solid parabolic tetrahedron
C        121              2         Rigid bar
C        122              32        Rigid element
C        136              2         Node to node translational spring
C        137              2         Node to node rotational spring
C        138              1         Node to ground translational spring
C        139              1         Node to ground rotational spring
C        141              2         Node to node damper
C        142              1         Node to ground damper
C        151              2         Node to node gap
C        152              1         Node to ground damper
C        161              1         Lumped mass
C        171              2         Axisymmetric linear shell
C        172              3         Axisymmetric parabolic shell
C        181              32        Constraint element
C
C ---- Dataset Organization
C
C      Record 1:      FORMAT(7I10)
C                     Field 1     -- Element lable
C                     Field 2     -- FE graphical description id
C                     Field 3     -- FE type id
C                     Field 4     -- Physical property table number
C                     Field 5     -- Material property table number
C                     Field 6     -- Color
C                     Field 7     -- Number of nodes on element
C
C      Record 2:      FORMAT(8I10)
C                     Field 1-n   -- Node lables defining element
C
C      Record 1 und 2 are repeated for each element in the model.
C
  140 CONTINUE
      READ (19, '(a80)', END=310) BUFFER
      IF (BUFFER(5:6) .NE. ('-1')) THEN
          ICOUNT = ICOUNT + 1
          READ (BUFFER, '(7I10)') NUM, IDGRPH, IDTYPE, IPHYS, MAT,
     1        ICOLOR, NNEL
          NPHYS(NUM) = IPHYS
C
C
          IF (IDTYPE.EQ.44 .OR. IDTYPE.EQ.54 .OR. IDTYPE.EQ.84) THEN
              ITYPE1 = 204
              ISTART = 1
              LINES = 4
              NIPXI = 2
              NIPETA = 2
              NIPSI = 0
              INTCOD = 0
```

```
          READ (19, '(8I10)') (NOP(I,NUM), I = 1, NNEL)
      ELSE IF(IDTYPE.EQ.45.OR.IDTYPE.EQ.55.OR.IDTYPE.EQ.85)THEN
          ITYPE1 = 208
          ISTART = 10
          LINES = 8
          NIPXI = 3
          NIPETA = 3
          NIPSI = 0
          INTCOD = 0
          READ (19, '(8I10)') NOP(1,NUM), NOP(5,NUM), NOP(2,NUM),
     1        NOP(6,NUM), NOP(3,NUM), NOP(7,NUM), NOP(4,NUM), NOP(8,
     2        NUM)
      ELSE IF (IDTYPE .EQ. 115) THEN
          ITYPE1 = 308
          ISTART = 18
          LINES = 12
          NIPXI = 2
          NIPETA = 2
          NIPSI = 2
          INTCOD = 0
          READ (19, '(8I10)') (NOP(I,NUM), I = 1, 8)
      ELSE IF (IDTYPE .EQ. 116) THEN
          ITYPE1 = 320
          ISTART = 30
          LINES = 24
          NIPXI = 3
          NIPETA = 3
          NIPSI = 3
          INTCOD = 0
          READ (19, '(8I10,2(/8I10))') NOP(1,NUM), NOP(9,NUM), NOP(2
     1        ,NUM), NOP(10,NUM), NOP(3,NUM), NOP(11,NUM), NOP(4,NUM
     2        ), NOP(12,NUM), NOP(13,NUM), NOP(14,NUM), NOP(15,NUM)
     3        , NOP(16,NUM), NOP(5,NUM), NOP(17,NUM), NOP(6,NUM),
     4        NOP(18,NUM), NOP(7,NUM), NOP(19,NUM), NOP(8,NUM), NOP(
     5        20,NUM)
      ELSE IF (IDTYPE .EQ. 94) THEN
          ITYPE1 = 304
          ISTART = 1
          LINES = 4
          NIPXI = 2
          NIPETA = 2
          NIPSI = 2
          INTCOD = 0
          READ (19, '(8I10)') NOP(1,NUM), NOP(2,NUM), NOP(3,NUM),
     1        NOP(4,NUM)
      ELSE IF (IDTYPE .EQ. 95) THEN
          ITYPE1 = 308
          ISTART = 10
          LINES = 8
          NIPXI = 2
          NIPETA = 2
          NIPSI = 2
          INTCOD = 0
          READ (19, '(8I10)') NOP(1,NUM), NOP(5,NUM), NOP(2,NUM),
     1        NOP(6,NUM), NOP(3,NUM), NOP(7,NUM), NOP(4,NUM), NOP(8,
     2        NUM)
      ELSE
          CALL ERRORS (9, 1, 'CAEDS ')
      ENDIF
C
C ---- Set bit 20 of ISPB to 1 to indicate that the element connect.
C      has been defined
C
```

```
          ISPB(NUM) = IBSET(ISPB(NUM),20)
C
          IF (IDTYPE.GE.40 .AND. IDTYPE.LT.50) THEN
              IFLAG = 1
          ELSE IF (IDTYPE.GE.50 .AND. IDTYPE.LT.60) THEN
              IFLAG = 2
          ELSE IF (IDTYPE.GE.80 .AND. IDTYPE.LT.90) THEN
              IFLAG = 3
          ELSE IF (IDTYPE.GE.90 .AND. IDTYPE.LT.100) THEN
              IFLAG = 4
          ENDIF
C
          INF1 = NNEL*8 + ITYPE1*512 + IFLAG*262144 + 2097152*NIPXI +
     1        16777216*NIPETA + 134217728*NIPSI + MAT
          INF2 = 8388608*ISTART + 131072*LINES + INTCOD
C
          INFOEL(1,NUM) = INF1
          INFOEL(2,NUM) = INF2
C
          LHIGH = MAX0(LHIGH,NUM)
          LLOW = MIN0(LLOW,NUM)
          GO TO 140
      ELSE
          WRITE (IOUT, 930) ICOUNT, ITYPE, LHIGH, LLOW
          IDSAT = 0
          GO TO 100
      ENDIF
C
C
C ---- Dataset 752:  Permanent groups
C
C
C ---- Dataset Organization
C
C       Record 1:       FORMAT(6I10)
C                       Field 1     -- group number
C                       Field 2     -- active constraint set number
C                       Field 3     -- active restraint set number
C                       Field 4     -- active load set number
C                       Field 5     -- active DOF set number
C                       Field 6     -- number of entities in group
C
C       Record 2:       FORMAT(20A2)
C                       Field 1     -- Group name
C
C       Record 3-N:     FORMAT(8I10)
C                       Field 1     -- entity type code
C                       Field 2     -- entity tag
C                       Field 3     -- entity type code
C                       Field 4     -- entity tag
C                       Field 5     -- entity type code
C                       Field 6     -- entity tag
C                       Field 7     -- entity type code
C                       Field 8     -- entity tag
C
C       Records 3 and 4 are repeated for each nodal force of the loadset
C
  150 CONTINUE
      READ (19, '(A80)', END=310) BUFFER
      IF (BUFFER(5:6) .NE. ('-1')) THEN
          READ (BUFFER, '(6I10)') IGROUP, I1, I2, I3, I4, NUMBER
          NFREE = MAX0(NFREE,IGROUP)
          READ (19, '(A40)') RSNAME
```

```
          IF (MOD(NUMBER,4) .EQ. 0) THEN
              K2 = NUMBER/4
          ELSE
              K2 = NUMBER/4 + 1
          ENDIF
C
          DO K1 = 1, K2
              READ (19, '(8I10)') (NVAR(K3), K3 = 1, 8)
              DO ID = 1, 7, 2
                  IF (NVAR(ID) .EQ. 8) THEN
                      ITAG = NVAR(ID+1)
                      IFBODY(ITAG) = IBSET(IFBODY(ITAG),IGROUP)
                  ENDIF
              END DO
          END DO
          WRITE (IOUT, 950) NUMBER, ITYPE, IGROUP, RSNAME
          GO TO 150
      ELSE
          IDSAT = 0
          GO TO 100
      ENDIF
C
C
C
C ---- DATASET 754:  CONSTRAINT SETS
C
C
C ---- DATASET ORGANIZATION
C
C
C
C       RECORD 1:      FORMAT(2I10)
C                      FIELD 1     -- CONSTRAINT SET NUMBER
C                      FIELD 2     -- CONSTRAINT TYPE
C                      FIELD 3        =0 - EMPTY SET
C                                     =1 - COUPLED DOFS
C                                     =2 - MULTI-POINT CONSTRAINTS
C
C       RECORD 2:      FORMAT(20A2)
C                      FIELD 1     -- CONSTRAINT SET NAME
C
C       FOR CONSTRAINT TYPE 1 - COUPLED DOFS
C
C       RECORD 3:      FORMAT(3I10,6I2)
C                      FIELD 1     -- INDEPENDENT NODE LABEL
C                      FIELD 2     -- COLOR NUMBER
C                      FIELD 3     -- NUMBER OF DEPENDENT NODES
C                      FIELD 4-9   -- SWITCHES FOR 1-6
C                                     =0 - OFF
C                                     =1 - ON
C
C       RECORD 4+N:    FORMAT(8I10)
C                      FIELD 1-8   -- DEPENDENT NODE LABELS
C
C
C    RECORD 3 AND 4 ARE REPEATED FOR EACH NODE IN THE CONSTRAINT SET.
C
C       FOR CONSTRAINT TYPE 2 - MULTI-POINT CONSTRAINTS
C
C       RECORD 3:      FORMAT(4I10,1P,2E13.5,I10)
C                      FIELD 1     -- EQUATION LABEL
C                      FIELD 2     -- NUMBER OF TERMS
C                      FIELD 3     -- FORCE/DISPLACEMENT SWITCH
```

```
C                                    =0 - FORCE
C                                    =1 - DISPLACEMENTS
C                     FIELD 4     -- COLOR
C                     FIELD 5     -- REAL PART OF FORCE/DISP. CONSTANT
C                     FIELD 6     -- IMAGINARY PART OF FOR./DISP. CONS.
C                     FIELD 7     -- DATA TYPE
C                                    =0 - REAL
C                                    =1 - COMPLEX
C
C       RECORD 4+N:   FORMAT(I10,I2,P1,2E13.5)
C                     FIELD 1     -- NODE LABELS
C                     FIELD 2     -- NODAL DOFS.
C                                  0 -- SCALAR
C                                  1 -- X DISP.
C                                  2 -- Y DISP.
C                                  3 -- Z DISP.
C                                  4 -- X ROT.
C                                  5 -- Y ROT.
C                                  6 -- Z ROT.
C                     FIELD 3     -- REAL PART OF CONSTRAINT COEFFTS.
C                     FIELD 4     -- NODE LABELS
C     RECORDS 3 AND 4+N ARE REPEATED FOR EACH CONSTRAINT SET.
C     THIS INCLUDES SEPARATORS,AND THE DATA SET TYPE RECORDS FOR
C     EACH TYPE OF EACH SET.
C
C
  180 CONTINUE
      READ (19, '(2I10)') NUM, IDTYPE
      READ (19, '(A40)') CSNAME
C
  190 CONTINUE
      READ (19, '(A80)', END=310) BUFFER
      IF (BUFFER(5:6) .NE. ('-1')) THEN
         IF (IDTYPE .EQ. 1) THEN
            READ (BUFFER, '(3I10,6I2)') INDL, ICOLOR, NDN, (ISW(I), I
     1           = 1, 6)
C
C       INDL -  INDEPENDENT NODE LABELS
C       NDN  -  NUMBER OF DEPENDENT NODES
C       ISW  -  SWITCHES FOR DEGREES OF FREEDOM.
C
            IF (MOD(NDN,8) .EQ. 0) THEN
               NREC = NDN/8
            ELSE
               NREC = NDN/8 + 1
            ENDIF
            IDENT1 = (INDL-1)*NNDF
            DO 220 K1 = 1, NREC
               READ (19, '(8I10)') (IDN(I), I = 1, 8)
               DO 210 K2 = 1, 8
                  IF (IDN(K2) .GT. 0) THEN
                     IDENT2 = (IDN(K2)-1)*NNDF
                     DO 200 K3 = 1, NNDF
                        IF (ISW(K3) .EQ. 1) THEN
                           ICOUNT = ICOUNT + 1
                           NMPC = NMPC + 1
                           MPCPNT = MPCPNT + 1
                           MPCADR(1,NMPC) = MPCPNT
                           MPCADR(2,NMPC) = 2
                           MPCDOF(MPCPNT) = IDENT1 + K3
                           COEFMP(MPCPNT) = 1.0
                           MPCPNT = MPCPNT + 1
                           MPCDOF(MPCPNT) = IDENT2 + K3
```

```
                           COEFMP(MPCPNT) = -1.0
                        ENDIF
  200                CONTINUE


                  ENDIF
  210             CONTINUE
  220          CONTINUE
            ELSE IF (IDTYPE .EQ. 2) THEN
               READ (BUFFER, '(4I10,1P,2E13.5,I10)') LEQ, NTERMS, ICASE,
     1            ICOLOR, CSTR, CSTI, IDATA
C
C    LEQ     EQUATION LABELS
C    NTERMS  NUMBER OF TERMS
C    ICASE   FORCE/DISP. SWITCH. 1 FORCE 2 DISPLACEMENTS
C    CSTR    CONST. PART OF EQUATION.
C    IDATA   DATA TYPE 0 REAL 1 COMPLEX
C
               ICOUNT = ICOUNT + 1
               NMPC = NMPC + 1
               MPCPNT = MPCPNT + 1
               MPCDOF(MPCPNT) = 0
               COEFMP(MPCPNT) = -CSTR
               MPCADR(1,NMPC) = MPCPNT
               MPCADR(2,NMPC) = NTERMS + 1
C
               DO 230 K1 = 1, NTERMS
                  READ (19, '(I10,I2,1P,2E13.5)') NODE, IDF, CSTR, CSTI
                  IF (IDF .LE. NNDF) THEN
                     IDENT1 = (NODE-1)*NNDF + IDF
                     MPCPNT = MPCPNT + 1
                     MPCDOF(MPCPNT) = IDENT1
                     COEFMP(MPCPNT) = CSTR
                  ELSE
                     CALL ERRORS (41, 1, 'CAEDS')
                  ENDIF
  230          CONTINUE
            ENDIF
            GO TO 190
         ELSE
            IDSAT = 0
            WRITE (IOUT, 960) ICOUNT, ITYPE, NUM, CSNAME
            GO TO 100
         ENDIF
C
C ---- Dataset 755:  Restraint Sets
C
C
C ---- Dataset Organization
C
C       Record 1:      FORMAT(2I10)
C                      Field 1     -- Restraint set number
C                      Field 2     -- Restraint type
C                      Field 3        =0 - empty set
C                                     =1 - nodal displacements
C                                     =2 - nodal temperature
C
C       Record 2:      FORMAT(20A2)
C                      Field 1     -- Restraint set name
C
C       For restraint type 1 - nodal displacements
C
C       Record 3:      FORMAT(2I10,7I2)
C                      Field 1     -- Restraint label
```

```
C                     Field 2     -- Color number
C                     Field 3-8   -- Switches for physical DOFs 1-6
C                                    =0 - off
C                                    =1 - on
C                     Field 9     -- Switch for user defined DOF
C                                    =0 - off
C                                    =1 - on
C
C       Record 4:     FORMAT(1P6,E13.5)
C                     Field 1-6   -- Displacement DOFs 1-6
C
C
C       Record 3 and 4 are repeated for each node in the restraint set.
C
  240 CONTINUE
      READ (19, '(2I10)') NUM, IDTYPE
      READ (19, '(A40)') RSNAME
      IF (IDTYPE .EQ. 2) CALL ERRORS (10, 1, 'CAEDS ')
C
  250 CONTINUE
      READ (19, '(A80)', END=310) BUFFER
      IF (BUFFER(5:6) .NE. ('-1')) THEN
          ICOUNT = ICOUNT + 1
          READ (BUFFER, '(2I10,7I2)') NODE, ICOLOR, (IDOF(I,NODE), I = 1
     1         , NNDF)
C
C ---- Identify the node as a support
C
          ISPB(NODE) = IBSET(ISPB(NODE),18)
          READ (19, '(1P,6E13.5)') (U(I,NODE), I = 1, NNDF)
          DO I = 1, NNDF
              IF (U(I,NODE) .NE. 0.0) IDOF(I,NODE) = -IDOF(I,NODE)
          END DO
          GO TO 250
      ELSE
          IDSAT = 0
          WRITE (IOUT, 970) ICOUNT, ITYPE, NUM, RSNAME
          GO TO 100
      ENDIF
C
C
C ---- Dataset 756:  Load sets
C
C
C ---- Dataset Organization
C
C       Record 1:     FORMAT(2I10)
C                     Field 1     -- Load set nymber
C                     Field 2     -- Load type
C                                    =0 - empty set
C                                    =1 - nodal force
C                                    =2 - nodal temperature
C                                    =3 - finite element face pressure
C                                    =4 - finite element face heat flux
C                                    =5 - finite element edge pressure
C                                    =6 - finite element edge heat flux
C                                    =7 - nodal heat source
C                                    =8 - finite element edge convect.
C                                    =9 - finite element edge radiation
C                                    =10- finite element face convect.
C                                    =11- finite element face radiation
C                                    =12- finite element dist. heat
C                                         generation
```

```
C                                  =13- finite element beam temp.
C                                  =14- load set acceleration data
C                                  =15- load set temperature data
C
C       Record 2:       FORMAT(20A2)
C                       Field 1      -- Load set name
C
C       For load type 1 - nodal forces
C
C       Record 3:       FORMAT(2I10,6I2)
C                       Field 1      -- nodal force label
C                       Field 2      -- Color number
C                       Field 3-8    -- Switches for DOFs 1-6
C                                       =0 - off
C                                       =1 - on
C
C       Record 4:       FORMAT(1P,6E13.5)
C                       Field 1-6    -- Forces for DOFs 1-6
C
C       Records 3 and 4 are repeated for each nodal force of the loadset
C
  270 CONTINUE
      READ (19, '(2I10)') LOADCN, IDTYPE
      READ (19, '(A40)') RSNAME
      IF (IDTYPE .NE. 1) CALL ERRORS (17, 1, 'CAEDS ')
C
  280 CONTINUE
      READ (19, '(A80)', END=310) BUFFER
      IF (BUFFER(5:6) .NE. ('-1')) THEN
          ICOUNT = ICOUNT + 1
          READ (BUFFER, '(2I10,7I2)') NODE, ICOLOR
          READ (19, '(1P,6E13.5)') (R(I,NODE), I = 1, NNDF)
          GO TO 280
      ELSE
          IDSAT = 0
          WRITE (IOUT, 980) ICOUNT, ITYPE, NUM, RSNAME
          GO TO 100
      ENDIF
C
  290 CONTINUE
      READ (19, '(2I10)') MODEL, NWORK
      READ (19, '(A80)') ID1
      READ (19, '(A80)', END=310) BUFFER
      IF (BUFFER(5:6) .EQ. ('-1')) THEN
          IDSAT = 0
          WRITE (IOUT, 940) ITYPE, MODEL, ID1
          GO TO 100
      ENDIF
C
C ---- Close the CAEDS universal file
C
  300 CONTINUE
      CLOSE(UNIT=19)
      NELEM = LHIGH
      NNODES = NHIGH
      RETURN
  310 CONTINUE
      CALL ERRORS (8, 1, 'CAEDS ')
  900           FORMAT(/1X,'------ A total of',I7,' records in',
     1                 ' dataset ',I3,' were skipped.')
  910       FORMAT(/1X,'------ A total of ',I6,' nodes in dataset ',I3,
     1           ' were processed.'/
     2           9X,'Highest Node Label = ',I6/
```

```
    3          9X,'Lowest Node Label  = ',I6)
  920     FORMAT(/1X,'------ A total of ',I6,' coordinate systems',
    1          ' in dataset ',I3,' were processed.')
  930     FORMAT(/1X,'------ A total of ',I6,' elements in dataset ',
    1          I3,' were processed.'/
    2          9X,'Highest Element Label = ',I6/
    3          9X,'Lowest Element Label  = ',I6)
  940     FORMAT(/1X,'------ A total of     2  records for',
    1          ' dataset ',I3,' were processed.'/
    2          9X,'Finite element model Number = ',I6/
    3          9X,'Finite Element model Name   = ',A80)
  950     FORMAT(/1X,'------ A total of ',I6,' entities in',
    1          ' dataset ',I3,' were processed.'/
    2          9X,'Group Number = ',I6/
    3          9X,'Group Name   = ',A40)
c changes made 7/18
  960     FORMAT(/1X,'------ A TOTAL OF ',I6,' CONSTRAINT FOR',
    1          ' DATASET ',I3,' WERE PROCESSED.'/
    2          9X,'CONSTRAINT SET NUMBER = ',I6/
    3          9X,'CONSTRAINT SET NAME   = ',A40)
c changes made 7/18
  970     FORMAT(/1X,'------ A total of ',I6,' restraint nodes for',
    1          ' dataset ',I3,' were processed.'/
    2          9X,'Restraint Set Number = ',I6/
    3          9X,'Restraint Set Name   = ',A40)
  980     FORMAT(/1X,'------ A Total of ',I6,' nodal loads for',
    1          ' dataset ',I6,' were processed.'/
    2          9X,'Load Set Number = ',I6/
    3          9X,'Load Set Name   = ',A40)
     END
C

C ========================= O U T 20 =============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE OUT20(IOUT,ICASE)
C ===================================================================
C I
C I   P R O G R A M:
C I
C I   OUT20 outputs various stresses and strains at the nodes on
C I   elements in the CAEDS universal file format.
C I
C I   A R G U M E N T   L I S T:
C I
C I   IOUT    = output device number of the NLRC3D output file
C I   ICASE   = 1: output second Piola-Kirchhoff stresses
C I           = 2: output Cauchy stresses
C I           = 3: output total strains
C I           = 4: output elastic strains
C I           = 5: output plastic strains
C I
C I
C I
C ==================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
```

```
C REWIN     ELINFO    ELINTM    LYINFO    LAGRG1    IOGET
C
C*************************************************************
      CHARACTER*1  CTEMP
      CHARACTER*80 ID1,ID2,ID3,ID4
      INTEGER ELNUM
      REAL*4 THICK
      REAL*8 LTHICK
      COMMON/UTIL1/VALUE(25),CTEMP
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1              NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC1(10),LDCONT,INCPTR
      COMMON/INPUTF/MATYPE(10)
      COMMON/ISPC01/P(64 , 27)
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/CAEDS1/I204(4),I208(8),I308(8),I320(20)
      COMMON/CAEDS2/ID1,ID2,ID3,ID4,LOADCN
      COMMON/INPUT9/THICK(9),IFLAG
      COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
C
C ---- Identification headers
C
C       ID1 = FE_model name
C       ID2 = run identification
C       ID3 = run date and time
C       ID4 = description of the data
C
      DIMENSION VALS(6,20),NID(20),MAXLYR(15)
      DATA ZERO/0.0D0/
C
C ---- Identify the output request
C
      ISTART = 0
      IEND = 0
      IVAL = 0
      IF (ICASE .EQ. 1) THEN
          ID4 = 'SECOND PIOLA KIRCHHOFF STRESSES'
          ISTART = 0
          IVAL = 2
      ELSE IF (ICASE .EQ. 3) THEN
          ID4 = 'TOTAL STRAINS'
          ISTART = 6
          IVAL = 3
      ELSE IF (ICASE .EQ. 4) THEN
          ID4 = 'ELASTIC STRAINS'
          ISTART = 12
          IVAL = 3
      ENDIF
C
      DO IM = 1, 15
          MAXLYR(IM) = 0
      END DO
      CALL REWIN
      DO 130 ELNUM = 1, NELEM
          CALL ELINFO (ELNUM, ITYPE, NNEL, IFLAG, IST, LINES)
          CALL ELINTM (ELNUM, IDENT, INTCOD, NIPXI, NIPETA, NIPSI,
     1        MATNUM, THICK)
C
          DO 120 LYNUM = 1, NLAYRS
              CALL LYINFO (LYNUM, LTHICK, ZS, MATRL, DCS, NNEL, ELNUM,
     1            NIPXI, NIPETA, NIPSI)
C ---- Write the universal dataset start code -1
```

```
C
            IF (MAXLYR(LYNUM) .EQ. 1) GO TO 110
            MAXLYR(LYNUM) = 1
            WRITE (20 + LYNUM, '(A6)') '    -1'
C
C ---- Write the dataset identification code 57
C
            WRITE (20 + LYNUM, '(A6)') '    57'
C
C ---- Write the five ID LINES. (empty at the moment)
C
C       ID1  = model identification
C       ID2  = run identification
C       ID3  = run date/time
C       ID4  = load case name
C       ID5  = (17H Load case number; I10)
C
            WRITE (20 + LYNUM, 900) ID1, ID2, ID3, ID4, LOADCN
C
C ---- Write record 6:   FORMAT(6,I10)
C
C       Field 1: model type        0:   unknown
C                                  1:   structural
C                                  2:   heat transfer
C                                  3:   fluid flow
C
C       Field 2: analysis type     0:   unknown
C                                  1:   static
C                                  2:   normal mode
C                                  3:   complex igenvalue
C                                  4:   transient
C                                  5:   frequency responce
C                                  6:   buckling
C
C       Field 3: data              0:   unknown
C                characteristic    2:   3 DOF global translation vec.
C                                  3:   6 DOF global trans. and rot. v.
C                                  4:   symmetric global tensor
C                                  5:   general global tensor
C                                  6:   shell and plate element stress
C                                       resultant
C
C       Field 4: specific data     0:   unknown
C                type (IVAL)       1:   unknown
C                                  2:   stress
C                                  3:   strain
C                                  4:   elelment force
C                                  5:   temperature
C                                  6:   heat flux
C                                  7:   strain energy
C                                  8:   displacement
C                                  9:   reaction force
C                                 10:   kinematic energy
C                                 11:   velocity
C                                 12:   acceleration
C                                 13:   strain energy density
C                                 14:   kinematic energy density
C                                 15:   hydrostatic pressure
C                                 16:   heat gradient
C                                 17:   code checking value
C                                 18:   coefficient of pressure
C                                 19:   ply stress
C                                 20:   ply strain
```

```
C                                 21:   failure index for ply
C                                 22:   failure index for bonding
C                                 23:   reaction heat flow
C                                 24:   stress error density
C                                 25:   stress variation
C                                 26:   scalar gradient
C                                 27:   shell and plate element stress
C                                       resultant
C
C       Field 5: data type        1:    integer
C                                 2:    single precision floating point
C                                 3:    double precision floating point
C                                 4:    single precision complex
C                                 5:    double precision complex
C
C       Field 6: number of data values for each position on the element
C
              WRITE (20 + LYNUM, 910) IVAL
C
C       Record 7 for analysis type =1, static FORMAT(8I10)
C
C       Field 1: 1
C       Field 2: 1


C       Field 3: Load case number
C
              WRITE (20 + LYNUM, 920)
C
C       Record 8 for analysis type =1, static format (6e13.5)
C
C       Field 1: 0.0
C
              WRITE (20 + LYNUM, '(E13.5)') ZERO
C
C       Records 9:           FORMAT(4I10)
C
C       Field 1:     Element number (ELNUM)
C       Field 2:     Data expansion code (IEXP)
C                    1: data present for all nodes
C                    2: data present for 1st node. all other nodes are
C                       the same
C       Field 3:     Number of nodes on element (NNEL)
C       Field 4:     Number of data values per node (NVPN)
C
C       Record 10:          FORMAT(6E13.5)
C
C       Feilds 1-n: data values at node 1 (NVPN real or complex valu.)
C
C          For IEXP = 1 record 10 is repeated NNEL times
C          FOR IEXP = 2 record 10 is repeated once.
C
C       Records 9 and 10 are repeated for all elements
C
  110         CONTINUE
  120     CONTINUE
  130 CONTINUE
      IEXP = 1
      NVPN = 6
      DO 280 ELNUM = 1, NELEM
          CALL ELINFO (ELNUM, ITYPE, NNEL, IFLAG, IST, LINES)
          CALL ELINTM (ELNUM, IDENT, INTCOD, NIPXI, NIPETA, NIPSI,
     1        MATNUM, THICK)
C
```

```
      DO 260 LYNUM = 1, NLAYRS
          CALL LYINFO (LYNUM, LTHICK, ZS, MATRL, DCS, NNEL, ELNUM,
     1        NIPXI, NIPETA, NIPSI)
          IF (ITYPE .LE. 0) GO TO 270
          IF (ITYPE.NE.0 .OR. IDENT.NE.0) THEN
              CALL LAGRG1 (ITYPE, IFLAG, NNEL, IERROR)
              IF (ITYPE .GT. 300) THEN
                  IF (IFLAG .EQ. 0) THEN
                      IF (ITYPE .EQ. 320) THEN
                          DO 140 K1 = 1, NNEL
                              NID(K1) = I320(K1)
  140                     CONTINUE
                      ELSE IF (ITYPE .EQ. 308) THEN
                          DO 150 K1 = 1, NNEL
                              NID(K1) = I308(K1)
  150                     CONTINUE
                      ENDIF
                  ELSE IF (IFLAG .EQ. 4) THEN
                      IF (ITYPE .EQ. 204) THEN
                          DO 160 K1 = 1, NNEL
                              NID(K1) = I204(K1)
  160                     CONTINUE
                      ELSE IF (ITYPE .EQ. 308) THEN
                          DO 170 K1 = 1, NNEL
                              NID(K1) = I208(K1)
  170                     CONTINUE
                      ENDIF
                  ENDIF
                  IEND = 6
              ELSE IF (ITYPE .GT. 200) THEN
                  IF (ITYPE .EQ. 204) THEN
                      DO 180 K1 = 1, NNEL
                          NID(K1) = I204(K1)
  180                 CONTINUE
                  ELSE IF (ITYPE .EQ. 208) THEN
                      DO 190 K1 = 1, NNEL
                          NID(K1) = I208(K1)
  190                 CONTINUE
                  ENDIF
                  IEND = 4
              ENDIF
          ENDIF
C
          DO 210 K1 = 1, NNEL
              DO 200 K2 = 1, IEND
                  VALS(K2,K1) = 0.
  200         CONTINUE
  210     CONTINUE
C
          DO 240 INTGPN = 1, NIP
              CALL IOGET (LDEV1, 96, '(A96)', 5)
              DO 230 K1 = 1, NNEL
                  CONST = P(INTGPN,K1)
                  DO 220 K2 = 1, IEND
c     if(LYNUM.NE.1) write(*,*)'steel stress',value(ISTART+K2),'p',const
                      VALS(K2,K1) = VALS(K2,K1) + VALUE(ISTART+K2)*
     1                    CONST
  220             CONTINUE
  230         CONTINUE
  240     CONTINUE
C
          WRITE (20 + LYNUM, '(4I10)') ELNUM, IEXP, NNEL, NVPN
C
```

```
C ---- Process expansion code 1
C
            DO 250 I = 1, NNEL
               ID = NID(I)
               IF (ITYPE .GT. 300) THEN
                  WRITE (20 + LYNUM, '(6E13.5)') VALS(1,ID), VALS(4,
     1                  ID), VALS(2,ID), VALS(6,ID), VALS(5,ID), VALS(
     2                  3,ID)
               ELSE IF (ITYPE .GT. 200) THEN
                  WRITE (20 + LYNUM, '(6E13.5)') VALS(1,ID), VALS(3,
     1                  ID), VALS(2,ID), ZERO, ZERO, VALS(4,ID)
               ENDIF
  250       CONTINUE
C
  260    CONTINUE
  270    CONTINUE
  280 CONTINUE
C
C ---- Close the universal dataset
C
      DO IM = 1, 15
         MAXLYR(IM) = 0
      END DO
      DO 320 ELNUM = 1, NELEM
         CALL ELINFO (ELNUM, ITYPE, NNEL, IFLAG, IST, LINES)
         CALL ELINTM (ELNUM, IDENT, INTCOD, NIPXI, NIPETA, NIPSI,
     1        MATNUM, THICK)
C
         DO 310 LYNUM = 1, NLAYRS
            CALL LYINFO (LYNUM, LTHICK, ZS, MATRL, DCS, NNEL, ELNUM,
     1           NIPXI, NIPETA, NIPSI)
C ---- Write the universal dataset start code -1
C
            IF (MAXLYR(LYNUM) .EQ. 1) GO TO 300
            MAXLYR(LYNUM) = 1
            WRITE (20 + LYNUM, '(A6)') '    -1'
  300       CONTINUE
  310    CONTINUE
  320 CONTINUE
      CALL REWIN
C
      RETURN
  900 FORMAT(4(A80/),'LOAD CASE NUMBER;',I10)
  910 FORMAT(9X,'1',9X,'1',9X,'4',I10,9X,'2',9X,'6')
  920 FORMAT(9X,'1',9X,'1',9X,'1')
      END
C
C ========================== O U T 2 1 ============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE OUT21(IOUT)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C************************************************************
      CHARACTER*80 ID1,ID2,ID3,ID4
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
```

```
     1              NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC1(10),LDCONT,INCPTR
      COMMON/INPUTE/ISPB(10000)
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/MAIN2/UTOTAL(60000)
      COMMON/CAEDS2/ID1,ID2,ID3,ID4,LOADCN
      DIMENSION DUMMY1( 6 )
      DATA ZERO/0.0D0/
C
C ---- Write the universal dataset start code -1
C
      WRITE (20, '(A6)') '    -1'
C
C ---- Write the dataset identification code 55
C
      WRITE (20, '(A6)') '    55'
C
C ---- Write the five ID LINES. (empty at the moment)
C
C       ID1  = model identification
C       ID2  = run identification
C       ID3  = run date/time
C       ID4  = description of results
C       ID5  = (17H Load case number; I10)
C
      IVAL = 0
      ID4 = 'NODAL DISPLACEMENTS'
      WRITE (20, 900) ID1, ID2, ID3, ID4, LOADCN
C
C ---- Write record 6:   FORMAT(6,I10)
C
C       Field 1: model type         0:   unknown
C                                   1:   structural
C                                   2:   heat transfer
C                                   3:   fluid flow
C
C       Field 2: analysis type      0:   unknown
C                                   1:   static
C                                   2:   normal mode
C                                   3:   complex igenvalue
C                                  -3:   complex igenvalue
C                                   4:   transient
C                                   5:   frequency responco
C                                   6:   buckling
C                                   7:   complex igenvalue ,second order
C
C       Field 3: data               0:   unknown
C                characteristic     1:   scalar
C                                   2:   3 DOF global translation vec.
C                                   3:   6 DOF global trans. and rot. v.
C                                   4:   symmetric global tensor
C                                   5:   general global tensor
C                                   6:   shell and plate element stress
C                                        resultant
C
C       Field 4: specific data      0:   unknown
C                type (IVAL)        1:   general
C                                   2:   stress
C                                   3:   strain
C                                   4:   elelment force
C                                   5:   temperature
C                                   6:   heat flux
C                                   7:   strain energy
```

```
C                                     8:    displacement
C                                     9:    reaction force
C                                    10:    kinematic energy
C                                    11:    velocity
C                                    12:    acceleration
C                                    13:    strain energy density
C                                    14:    kinematic energy density
C                                    15:    hydrostatic pressure
C                                    16:    heat gradient
C                                    17:    code checking value
C                                    18:    coefficient of pressure
C                                    19:    ply stress
C                                    20:    ply strain
C                                    21:    failure index for ply
C                                    22:    failure index for bonding
C                                    23:    reaction heat flow
C                                    24:    stress error density
C                                    25:    stress variation
C                                    26:    scalar gradient
C                                    27:    shell and plate element stress
C                                           resultant
C
C       Field 5: data type           1:    integer
C                                    2:    single precision floating point
C                                    4:    double precision floating point
C                                    5:    single precision complex
C                                    6:    double precision complex
C
C       Field 6: number of data values for each position on the element
C
      WRITE (20, 910) IVAL
C
C       Record 7 for analysis type =1, static FORMAT(8I10)
C
C       Field 1: 1
C       Field 2: 1
C       Field 3: Load case number
C
      WRITE (20, 920)
C
C       Record 8 for analysis type =1, static format (6e13.5)
C
C       Field 1: 0.0
C
      WRITE (20, '(E13.5)') ZERO
C
C       Records 9:          FORMAT(I10)
C
C       Field 1:    node number (NODE)
C
C       Record 10:         FORMAT(6E13.5)
C
C       Feilds 1-n: data values at node 1 (NDV real or complex valu.)
C
C       Records 9 and 10 are repeated for all elements
C
      DO 100 K1 = 1, NNODES
         IF (BTEST(ISPB(K1),10)) THEN
            ID = NNDF*(K1-1)
            WRITE (20, 930) K1, (UTOTAL(ID+K2), K2 = 1, NNDF)
         ENDIF
  100 CONTINUE
C
```

```
C
C ---- Close the universal dataset
C
      WRITE (20, '(A6)') '    -1'
      RETURN
  900 FORMAT(4(A80/),'LOAD CASE NUMBER;',I10)
  910 FORMAT(9X,'1',9X,'1',9X,'3',9X,'8',9x,'2',9X,'6')
  920 FORMAT(9X,'1',9X,'1',9X,'1')
  930 FORMAT(I10/6E13.5)
      END
C
C ========================= O U T 2 2 ===========================
C
C     INCLUDE (PROCESS)
      SUBROUTINE OUT22(IOUT)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C***********************************************************
      CHARACTER*80 ID1,ID2,ID3,ID4
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1              NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC1(10),LDCONT,INCPTR
      COMMON/INPUTE/ISPB(10000)
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/MAIN4/RE(60000)
      COMMON/CAEDS2/ID1,ID2,ID3,ID4,LOADCN
      DIMENSION DUMMY1( 6 )
      DATA ZERO/0.0D0/
C
C ---- Write the universal dataset start code -1
C
      WRITE (20, '(A6)') '    -1'
C
C ---- Write tho dataset identification code 55
C
      WRITE (20, '(A6)') '    55'
C
C ---- Write the five ID LINES. (empty at the moment)
C
C       ID1  = model identification
C       ID2  = run identification
C       ID3  = run date/time
C       ID4  = description of results
C       ID5  = (17H Load case number; I10)
C
      IVAL = 0
      ID4 = 'SUPPORT REACTIONS'
      WRITE (20, 900) ID1, ID2, ID3, ID4, LOADCN
C
C ---- Write record 6:   FORMAT(6,I10)
C
C       Field 1: model type          0:   unknown
C                                    1:   structural
C                                    2:   heat transfer
C                                    3:   fluid flow
C
```

```
C      Field 2: analysis type      0:    unknown
C                                  1:    static
C                                  2:    normal mode
C                                  3:    complex igenvalue
C                                 -3:    complex igenvalue
C                                  4:    transient
C                                  5:    frequency responce
C                                  6:    buckling
C                                  7:    complex igenvalue ,second order
C
C      Field 3: data               0:    unknown
C               characteristic     1:    scalar
C                                  2:    3 DOF global translation vec.
C                                  3:    6 DOF global trans. and rot. v.
C                                  4:    symmetric global tensor
C                                  5:    general global tensor
C                                  6:    shell and plate element stress
C                                        resultant
C
C      Field 4: specific data      0:    unknown
C               type (IVAL)        1:    general
C                                  2:    stress
C                                  3:    strain
C                                  4:    elelment force
C                                  5:    temperature
C                                  6:    heat flux
C                                  7:    strain energy
C                                  8:    displacement
C                                  9:    reaction force
C                                 10:    kinematic energy
C                                 11:    velocity
C                                 12:    acceleration
C                                 13:    strain energy density
C                                 14:    kinematic energy density
C                                 15:    hydrostatic pressure
C                                 16:    heat gradient
C                                 17:    code checking value
C                                 18:    coefficient of pressure
C                                 19:    ply stress
C                                 20:    ply strain
C                                 21:    failure index for ply
C                                 22:    failure index for bonding
C                                 23:    reaction heat flow
C                                 24:    stress error density


C                                 25:    stress variation
C                                 26:    scalar gradient
C                                 27:    shell and plate element stress
C                                        resultant
C
C      Field 5: data type          1:    integer
C                                  2:    single precision floating point
C                                  4:    double precision floating point
C                                  5:    single precision complex
C                                  6:    double precision complex
C
C      Field 6: number of data values for each position on the element
C
      WRITE (20, 910) IVAL
C
C      Record 7 for analysis type =1, static FORMAT(8I10)
C
C      Field 1: 1
```

```
C       Field 2: 1
C       Field 3: Load case number
C
      WRITE (20, 920)
C
C       Record 8 for analysis type =1, static format (6e13.5)
C
C       Field 1: 0.0
C
      WRITE (20, '(E13.5)') ZERO
C
C       Records 9:          FORMAT(I10)
C
C       Field 1:    node number (NODE)
C
C       Record 10:         FORMAT(6E13.5)
C
C       Feilds 1-n: data values at node 1 (NDV real or complex valu.)
C
C       Records 9 and 10 are repeated for all elements
C
C
      DO 100 K1 = 1, NNODES
          IF (BTEST(ISPB(K1),18)) THEN
              ID = NNDF*(K1-1)
              WRITE (20, 930) K1, (RE(ID+K2), K2 = 1, NNDF)
          ENDIF
  100 CONTINUE
C
C ---- close the universal dataset
C
      WRITE (20, '(A6)') '    -1'
      RETURN
  900 FORMAT(4(A80/),'LOAD CASE NUMBER;',I10)
  910 FORMAT(9X,'1',9X,'1',9X,'3',9X,'9',9x,'2',9X,'6')
  920 FORMAT(9X,'1',9X,'1',9X,'1')
  930 FORMAT(I10/6E13.5)
      END
C     INCLUDE(PROCESS)
      SUBROUTINE RESTAT
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C**************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ELINFO    ELINTM    LYINFO
C
C**************************************************************
      REAL*4  THICK
      REAL*8  LTHICK
      INTEGER ELNUM,ELNUMB
      COMMON/LPROP/PROPER(25,15)
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC1,MNIT,IFLAG1,IFLAG2,IDIM,
     $              NINODE,NCOLOR,NFREE
      COMMON/INPUT9/THICK(9),IFLAG
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
      COMMON/ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON/INCR11/FRACT(10),NLINC(10),LDCONT,INCPTR
```

```
      COMMON/TSHEAR/AK1(5000,9),AK2(5000,9),AF1(5000,15,27)
     $,AF2(5000,15,27)
C
C
C        SUBROUTINE CONTROLS THE RESTART PROCEDURE
C        AND SETS UP THE APPROPRIATE ARRAYS FOR CONCRETE AND STEEL
C
C CRACK(L,I,J) = ITH CRACK DATA AT LTH SAMPLE POINT OF JTH LAYER
C FOR EACH ELEMENT
C HISTO(L,I,J) = ITH MATER. HIST. DATA AT LTH SAMPLE POINT OF JTH LAYER
C FOR EACH ELEMENT
C TENSTF(L,I,J) = ITH TENS. STIF. DATA AT LTH SAMPLE POINT OF JTH LAYER
C FOR EACH ELEMENT(CONCRETE ONLY)
C IHISTY,IHIST1,IHIST2 ARE FILES USED FOR STOREING ARRAY CRACK
C ITNSPG,ITNSG1,ITNSG2 ARE FILES USED FOR STOREING ARRAY HISTO,TENSTF
C
C=====================================================================
C
      IHISTY = 50
      IHIST1 = 60
      IHIST2 = 61
      ITNSPG = 80
      ITNSG1 = 70
      ITNSG2 = 71
      ICP = 62
      REWIND IHISTY
      REWIND IHIST1
      REWIND IHIST2
      REWIND ITNSPG
      REWIND ITNSG1
      REWIND ITNSG2
      REWIND ICP
      DO 110 ELNUM = 1, NELEM
C
          CALL ELINFO (ELNUM, ITYPE, NNEL, IFLAG, ISTART, LINES)
          CALL ELINTM (ELNUM, IDENT, INTCODE, NIPXI, NIPETA, NIPSI,
     1        MATNUM, THICK)
C
          DO 100 LYNUM = 1, NLAYRS
C
              CALL LYINFO (LYNUM, LTHICK, ZS, MATRL, DCS, NNEL, ELNUM,
     1            NIPXI, NIPETA, NIPSI)
C
C        WRITE(*,*) 'NELEM',NELEM,'NLAYRS',NLAYRS
C
C  RECOVER ARRAY 'HISTO','CRACK','TENSTF' FOR THE CURRENT ELEMENT.
C  THESE ARE REQUIRED FOR THE UPDATING PROCEDURE IN THE CONVERGENCE
C  CRITERION EMPLOYED. STORES ALL THE MATERIAL INFO ABOUT POINT.
C
              NIP = NIPXI*NIPETA*NIPSI
C
              READ (IHIST2) ELNUMB, LYNUMB, NGAUS, ((CRACK(L,K,LYNUM),K
     1             = 1,250), L = 1, NGAUS)
C
              IF (ELNUMB.NE.ELNUM .OR. LYNUMB.NE.LYNUM .OR. NGAUS.NE.NIP
     1            ) THEN
                  WRITE (*, *) 'ERROR READING IHISTY IN RESTAT'
                  WRITE (*, *) 'ELNUM', ELNUM, 'ELNUMB', ELNUMB, 'LYNUM'
     1                , LYNUM, 'LYNUMB', LYNUMB, 'NIP', NIP, 'NGAUS',
     2                NGAUS
                  STOP
              ENDIF
C
```

```
            WRITE (IHISTY) ELNUMB, LYNUMB, NGAUS, ((CRACK(L,K,LYNUM),K
     1           = 1,250), L = 1, NGAUS)
C
            WRITE (IHIST1) ELNUMB, LYNUMB, NGAUS, ((CRACK(L,K,LYNUM),K
     1           = 1,250), L = 1, NGAUS)
C
            READ (ITNSG2) ELNUMB, LYNUMB, NGAUS, MATRL, ((TENSTF(L,K,
     1          LYNUM),K = 1,80), L = 1, NGAUS), ((HISTO(L,K,LYNUM),K
     2           = 1,70), L = 1, NGAUS), (PROPER(KM,MATRL), KM = 1, 25
     3          )
C
            IF (ELNUMB.NE.ELNUM .OR. LYNUMB.NE.LYNUM .OR. NGAUS.NE.NIP
     1          ) THEN
                WRITE (*, *) 'ERROR READING ITNSPG IN RESTAT'
                WRITE (*, *) 'ELNUM', ELNUM, 'ELNUMB', ELNUMB, 'LYNUM'
     1              , LYNUM, 'LYNUMB', LYNUMB, 'NIP', NIP, 'NGAUS',
     2              NGAUS
                STOP
            ENDIF
C
            WRITE (ITNSPG) ELNUMB, LYNUMB, NGAUS, MATRL, ((TENSTF(L,K,
     1          LYNUM),K = 1,80), L = 1, NGAUS), ((HISTO(L,K,LYNUM),K
     2           = 1,70), L = 1, NGAUS), (PROPER(KM,MATRL), KM = 1, 25
     3          )
C
            WRITE (ITNSG1) ELNUMB, LYNUMB, NGAUS, MATRL, ((TENSTF(L,K,
     1          LYNUM),K = 1,80), L = 1, NGAUS), ((HISTO(L,K,LYNUM),K
     2           = 1,70), L = 1, NGAUS), (PROPER(KM,MATRL), KM = 1, 25
     3          )
C
            READ (ICP) ELNUMB, LYNUMB, NGAUS, (AF1(ELNUM,LYNUM,KJ), KJ
     1           = 1, NIP), (AF2(ELNUM,LYNUM,KP), KP = 1, NIP)
C
            IF (ELNUMB.NE.ELNUM .OR. LYNUMB.NE.LYNUM .OR. NGAUS.NE.NIP
     1          ) THEN
                WRITE (*, *) 'ERROR READING IC2 IN RESTAT'
                WRITE (*, *) 'ELNUM', ELNUM, 'ELNUMB', ELNUMB, 'LYNUM'
     1              , LYNUM, 'LYNUMB', LYNUMB, 'NIP', NIP, 'NGAUS',
     2              NGAUS
                STOP
            ENDIF
  100     CONTINUE
C
  110 CONTINUE
C
      RETURN
      END

C*************************************************************
C     INCLUDE(PROCESS)
      SUBROUTINE CLEN(NREL,INTGPN,ITYPE,NNEL,IERROR,TDCS,CL)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ISOP3D    N2D       COORD1    SIMUL
C
C*************************************************************
C
C     TO EVALUATE THE CONSISTANT CHARACTERISTIC LENGTH OF A CRACKED
```

```
C     SAMPLE POINT IN A CONCRETE LAYER.
C     CALLED BY CKINIT
C
      REAL*4 THICK,XYZ
      REAL*8 N,NXI,NETA,NSI
      COMMON/INPUT9/THICK(9),IFLAG
      COMMON/CONSIS/RSTLOC(27,3)
      COMMON/INPUT2/NOP(20,5000)
      COMMON/TRANS/ DC(3,3)
      COMMON/INPUT3/XYZ(3,10000)
      DIMENSION PHI(8),TDCS(3,3),XTMP(3,8),CX(3),N(20),NXI(20),
     $NETA(20),NSI(20),C(8,8),TEMP(4,8),XTMP1(3,8)
C
      CL = 0.0
      NEL1 = 0
      DO JM = 1, 8
          DO JK = 1, 8
              C(JM,JK) = 0.0
          END DO
          TEMP(1,JM) = 0.0
          TEMP(2,JM) = 0.0
          TEMP(3,JM) = 0.0
          TEMP(4,JM) = 0.0
      END DO
C
C COMPUTE THE COORDINATES OF THE CENTER OF THE ELEMENT(LOCAL 0,0,0)
C
      IF (ITYPE.GE.300 .AND. IFLAG.EQ.0) THEN
          NEL1 = 8
          XIC = 0.0
          ETAC = 0.0
          SIC = 0.0
          ITYPE1 = 308
          CALL ISOP3D(XIC,ETAC,SIC,N,NXI,NETA,NSI,ITYPE1,IERROR)
          CX(1) = 0.0
          CX(2) = 0.0
          CX(3) = 0.0
          DO 120 I = 1, 8
              NODE = NOP(I,NREL)
              CX(1) = CX(1) + N(I)*XYZ(1,NODE)
              CX(2) = CX(2) + N(I)*XYZ(2,NODE)
              CX(3) = CX(3) + N(I)*XYZ(3,NODE)
 120      CONTINUE
C
C           LOCAL NODAL COORDINATES
C
          DO 130 II = 1, 8
              XTMP(1,II) = XYZ(1,NOP(II,NREL)) - CX(1)
              XTMP(2,II) = XYZ(2,NOP(II,NREL)) - CX(2)
              XTMP(3,II) = XYZ(3,NOP(II,NREL)) - CX(3)
 130      CONTINUE
      ELSE IF (ITYPE.GE.300 .AND. IFLAG.EQ.4) THEN
C           WRITE(*,*)'SHELL'
          NEL1 = 4
          XIC = 0.0
          ETAC = 0.0
          SIC = 0.0
          ITYPE1 = 204
          CALL N2D (XIC, ETAC, N, ITYPE1, IERROR)
          CX(1) = 0.0
          CX(2) = 0.0
          CX(3) = 0.0
          NODE = NOP(1,NREL)
```

```
         CX(1) = CX(1) + N(1)*XYZ(1,NODE)
         CX(2) = CX(2) + N(1)*XYZ(2,NODE)
         CX(3) = CX(3) + N(1)*XYZ(3,NODE)
         NODE = NOP(2,NREL)
         CX(1) = CX(1) + N(2)*XYZ(1,NODE)
         CX(2) = CX(2) + N(2)*XYZ(2,NODE)
         CX(3) = CX(3) + N(2)*XYZ(3,NODE)
         NODE = NOP(3,NREL)
         CX(1) = CX(1) + N(3)*XYZ(1,NODE)
         CX(2) = CX(2) + N(3)*XYZ(2,NODE)
         CX(3) = CX(3) + N(3)*XYZ(3,NODE)
         NODE = NOP(4,NREL)
         CX(1) = CX(1) + N(4)*XYZ(1,NODE)
         CX(2) = CX(2) + N(4)*XYZ(2,NODE)
         CX(3) = CX(3) + N(4)*XYZ(3,NODE)
C
C          LOCAL NODAL COORDINATES
C
         XTMP1(1,1) = XYZ(1,NOP(1,NREL)) - CX(1)
         XTMP1(2,1) = XYZ(2,NOP(1,NREL)) - CX(2)
         XTMP1(3,1) = XYZ(3,NOP(1,NREL)) - CX(3)
         XTMP1(1,2) = XYZ(1,NOP(2,NREL)) - CX(1)
         XTMP1(2,2) = XYZ(2,NOP(2,NREL)) - CX(2)
         XTMP1(3,2) = XYZ(3,NOP(2,NREL)) - CX(3)
         XTMP1(1,3) = XYZ(1,NOP(3,NREL)) - CX(1)
         XTMP1(2,3) = XYZ(2,NOP(3,NREL)) - CX(2)
         XTMP1(3,3) = XYZ(3,NOP(3,NREL)) - CX(3)
         XTMP1(1,4) = XYZ(1,NOP(4,NREL)) - CX(1)
         XTMP1(2,4) = XYZ(2,NOP(4,NREL)) - CX(2)
         XTMP1(3,4) = XYZ(3,NOP(4,NREL)) - CX(3)
         DO 150 II = 1, 4
             DO IP = 1, 3
                 XTMP(IP,II) = 0.0
                 XTMP(IP,II) = XTMP(IP,II) + DC(1,IP)*XTMP1(1,II)
                 XTMP(IP,II) = XTMP(IP,II) + DC(2,IP)*XTMP1(2,II)
                 XTMP(IP,II) = XTMP(IP,II) + DC(3,IP)*XTMP1(3,II)
             END DO
  150    CONTINUE
     ENDIF
C
C FIND THE PHI VALUES
C
     AL1 = TDCS(1,1)
     AM1 = TDCS(1,2)
     AN1 = TDCS(1,3)
     IF (ITYPE.GE.300 .AND. IFLAG.EQ.4) THEN
         AL1=DC(1,1)*TDCS(1,1)+DC(2,1)*TDCS(1,2)+DC(3,1)*TDCS(1,3)
         AM1=DC(1,2)*TDCS(1,1)+DC(2,2)*TDCS(1,2)+DC(3,2)*TDCS(1,3)
         AN1=DC(1,3)*TDCS(1,1)+DC(2,3)*TDCS(1,2)+DC(3,3)*TDCS(1,3)
     ENDIF
     DO 160 I = 1, NEL1
         XROT = AL1*XTMP(1,I) + AM1*XTMP(2,I) + AN1*XTMP(3,I)
         PHI(I) = 0.0
         IF (XROT .GE. 0.) PHI(I) = 1.0
  160 CONTINUE
C
C FIND THE DERIVATIVES OF THE SHAPE FUNCTIONS(LINEAR ORDER) @ INTGPN
C
     X1 = 0.0
     Y1 = 0.0
     Z1 = 0.0
     CALL COORD1 (NREL, NNEL, INTGPN, ITYPE, X1, Y1, Z1)
     IF (ITYPE.GE.300 .AND. IFLAG.EQ.4) THEN
```

```
            X1 = X1 - CX(1)
            Y1 = Y1 - CX(2)
            Z1 = Z1 - CX(3)
            X2 = DC(1,1)*X1 + DC(2,1)*Y1 + DC(3,1)*Z1
            Y2 = DC(1,2)*X1 + DC(2,2)*Y1 + DC(3,2)*Z1
            Z2 = DC(1,3)*X1 + DC(2,3)*Y1 + DC(3,3)*Z1
            X1 = X2
            Y1 = Y2
            Z1 = Z2
        ENDIF
        IF (ITYPE.GE.300 .AND. IFLAG.EQ.0) THEN
            NEL1 = 8
            DO 170 IM = 1, 8
                C(IM,1) = 1.0
                C(IM,2) = XYZ(1,NOP(IM,NREL))
                C(IM,3) = XYZ(2,NOP(IM,NREL))
                C(IM,4) = XYZ(3,NOP(IM,NREL))
                C(IM,5) = XYZ(1,NOP(IM,NREL))*XYZ(2,NOP(IM,NREL))
                C(IM,6) = XYZ(2,NOP(IM,NREL))*XYZ(3,NOP(IM,NREL))
                C(IM,7) = XYZ(3,NOP(IM,NREL))*XYZ(1,NOP(IM,NREL))
                C(IM,8) = XYZ(1,NOP(IM,NREL))*XYZ(2,NOP(IM,NREL))*XYZ(3,
     1              NOP(IM,NREL))
  170       CONTINUE
C
            NROW = 8
            NRCOL = 8
            EPS = 1.E-20
            DETER = SIMUL(NRCOL,C,EPS,NROW,IOUT)
            IF (DETER .EQ. 0) WRITE (*, *) 'ERROR IN SIMUL SUB-CLEN'
            TEMP(1,1) = 1.0
            TEMP(1,2) = X1
            TEMP(1,3) = Y1
            TEMP(1,4) = Z1
            TEMP(1,5) = X1*Y1
            TEMP(1,6) = Y1*Z1
            TEMP(1,7) = Z1*X1
            TEMP(1,8) = X1*Y1*Z1
            TEMP(2,2) = 1.0
            TEMP(2,5) = Y1
            TEMP(2,7) = Z1
            TEMP(2,8) = Y1*Z1
            TEMP(3,3) = 1.0
            TEMP(3,5) = X1
            TEMP(3,6) = Z1
            TEMP(3,8) = X1*Z1
            TEMP(4,4) = 1.0
            TEMP(4,6) = Y1
            TEMP(4,7) = X1
            TEMP(4,8) = X1*Y1
C
            DO 190 IM = 1, 8
                N(IM) = 0.0
                NXI(IM) = 0.0
                NETA(IM) = 0.0
                NSI(IM) = 0.0
                DO 180 JK = 1, 8
                    N(IM) = N(IM) + TEMP(1,JK)*C(JK,IM)
                    NXI(IM) = NXI(IM) + TEMP(2,JK)*C(JK,IM)
                    NETA(IM) = NETA(IM) + TEMP(3,JK)*C(JK,IM)
                    NSI(IM) = NSI(IM) + TEMP(4,JK)*C(JK,IM)
  180           CONTINUE
  190       CONTINUE
        ELSE IF (ITYPE.GE.300 .AND. IFLAG.EQ.4) THEN
```

```
        NEL1 = 4

        C(1,1) = 1.0
        C(1,2) = XTMP(1,1)
        C(1,3) = XTMP(2,1)
        C(1,4) = XTMP(1,1)*XTMP(2,1)
        C(2,1) = 1.0
        C(2,2) = XTMP(1,2)
        C(2,3) = XTMP(2,2)
        C(2,4) = XTMP(1,2)*XTMP(2,2)
        C(3,1) = 1.0
        C(3,2) = XTMP(1,3)
        C(3,3) = XTMP(2,3)
        C(3,4) = XTMP(1,3)*XTMP(2,3)
        C(4,1) = 1.0
        C(4,2) = XTMP(1,4)
        C(4,3) = XTMP(2,4)
        C(4,4) = XTMP(1,4)*XTMP(2,4)
        NROW = 8
        NRCOL = 4
        EPS = 1.E-20
        DETER = SIMUL(NRCOL,C,EPS,NROW,IOUT)
        IF (DETER .EQ. 0) WRITE (*, *) 'ERROR IN SIMUL SUB-CLEN'
        TEMP(1,1) = 1.0
        TEMP(1,2) = X1
        TEMP(1,3) = Y1
        TEMP(1,4) = X1*Y1
        TEMP(2,2) = 1.0
        TEMP(2,4) = Y1
        TEMP(3,3) = 1.0
        TEMP(3,4) = X1
C
        DO 200 IM = 1, 4
           N(IM) = 0.0
           NXI(IM) = 0.0
           NETA(IM) = 0.0
           NSI(IM) = 0.0
           N(IM) = N(IM) + TEMP(1,1)*C(1,IM)
           NXI(IM) = NXI(IM) + TEMP(2,1)*C(1,IM)
           NETA(IM) = NETA(IM) + TEMP(3,1)*C(1,IM)
           N(IM) = N(IM) + TEMP(1,2)*C(2,IM)
           NXI(IM) = NXI(IM) + TEMP(2,2)*C(2,IM)
           NETA(IM) = NETA(IM) + TEMP(3,2)*C(2,IM)
           N(IM) = N(IM) + TEMP(1,3)*C(3,IM)
           NXI(IM) = NXI(IM) + TEMP(2,3)*C(3,IM)
           NETA(IM) = NETA(IM) + TEMP(3,3)*C(3,IM)
           N(IM) = N(IM) + TEMP(1,4)*C(4,IM)
           NXI(IM) = NXI(IM) + TEMP(2,4)*C(4,IM)
           NETA(IM) = NETA(IM) + TEMP(3,4)*C(4,IM)
  200   CONTINUE
     ENDIF
     SUM = 0.0
     DO 210 K = 1, NEL1
        SUM = SUM + (NXI(K)*AL1+NETA(K)*AM1+NSI(K)*AN1)*PHI(K)
  210 CONTINUE
C
     CL = 1.0/SUM
C
     RETURN
     END
C ---------------------------  C K I N I T  ----------------------
C
C    INCLUDE(PROCESS)
```

```
      SUBROUTINE CKINIT(L,J,PROPER,IELEM,NNEL,IOUT,ITYPE,DTSTIF,IERROR,
     $IFLAG1,IFLAG2,IFLAG3,ITCRK,ICOUP,IFLAG8)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C CLEN      INITES    SIMUL     DTRANS
C
C************************************************************
      REAL*8 LTHICK
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON/ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      COMMON/MATER1/DEP(6,6)
      COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),MLAYRS,MATRL,LYNUM
      COMMON/ICKPIF/ICKFGO,ICKNTR
      DIMENSION PROPER(25),DOT(3),ROT(3,3)
      REAL*8 NUS,NUSFLR,N(3,3),NT(3,3),TENXYZ(6,6),DTSTIF(6,6)
C
C   FLAG ON TO STORE CRACK ORIENT INFO FOR CRACKPLOTTING
C
      ICKFGO = 1
      EPS1 = 0.0
      EPS2 = 0.0
      EPS3 = 0.0
      GF = 0.0
      SIGF = 0.0
      EPSNOT = 0.0
      COFET = 0.0
C
C              SUBROUTINE TO REGISTER CRACK INITIATION PARAMETERS.
C
C             ...   INCREASE NUMBER OF CRACKS BY ONE.
C
      CRACK(L,1,J) = CRACK(L,1,J) + 1.
C
C              GET NO. OF CRACKS AS AN INTEGER.
C
      NCRACK = INT(CRACK(L,1,J))
C
C
C        RECORD THE DIRECTION COSINES OF THE CRACK NORMAL WITH RESPECT
C        TO THE XYZ COORDINATE SYSTEM.( SAME AS THE MAX. PRINCIPAL
C        DIRECTION).
C
      NCK = 9*(NCRACK-1) + 1
      CRACK(L,137+NCK,J) = HISTO(L,10,J)
      ROT(1,1) = HISTO(L,10,J)
      CRACK(L,138+NCK,J) = HISTO(L,11,J)
      ROT(1,2) = HISTO(L,11,J)
      CRACK(L,139+NCK,J) = HISTO(L,12,J)
      ROT(1,3) = HISTO(L,12,J)
      CRACK(L,140+NCK,J) = HISTO(L,13,J)
      ROT(2,1) = HISTO(L,13,J)
      CRACK(L,141+NCK,J) = HISTO(L,14,J)
      ROT(2,2) = HISTO(L,14,J)
      CRACK(L,142+NCK,J) = HISTO(L,15,J)
      ROT(2,3) = HISTO(L,15,J)
      CRACK(L,143+NCK,J) = HISTO(L,16,J)
      ROT(3,1) = HISTO(L,16,J)
```

```
      CRACK(L,144+NCK,J) = HISTO(L,17,J)
      ROT(3,2) = HISTO(L,17,J)
      CRACK(L,145+NCK,J) = HISTO(L,18,J)
      ROT(3,3) = HISTO(L,18,J)
C
C             STORE THE STRESS AT WHICH THE CRACK INITIATED.
C
      SIGM1F = DMAX1(DMAX1(HISTO(L,34,J),HISTO(L,35,J)),HISTO(L,36,J))
      CRACK(L,37+NCRACK,J) = SIGM1F
C
C               INITIALIZE THE STATE OF CRACK TO LOADING SITUATION
C               I.E.,INCREASE IN STRAIN.
C
      CRACK(L,117+NCRACK,J) = 1.
C
C                IF THE CRACK IS RELATED TO TENSION-STIFFENING ...
C
      IF (PROPER(16) .GT. 0.0) THEN
C
C                FOR THE TIME BEING SET THE TERMINATING STRAIN TO
C                THAT ASSIGNED EXTERNALLY.
C
          CRACK(L,127+NCRACK,J) = PROPER(17)
          GO TO 120
      ENDIF
C
C CALL CRACK BAND WIDTH ROUTINE TO EVALUATE THE CRACK WIDTH AT SAMPLE
C  POINT (BASED ON THE WORK OF OLIVER IJNME 1989.)
C
      IF (DABS(ROT(1,1)).GE.0.707 .AND. DABS(ROT(1,1)).LE.0.708) THEN
          ROT(1,1) = ROT(1,1) + 0.01
          ROT(1,2) = ROT(1,2) - 0.01
          ROT(1,3) = ROT(1,3) - 0.01
      ENDIF
      CALL CLEN (IELEM, L, ITYPE, NNEL, IERROR, ROT, CL)
C
  100 CONTINUE
      GF = PROPER(11)
C
C       THE FRACTURE ENERGY RELEASE AS A FUNCTION OF THE CRACK
C       ORIENTATION WITH RESPECT TO THE REINFORCEMENT IS ESTIMATED.
C
      DO 110 JKL = 1, 3
          IF (TENSTF(L,3+JKL,J) .GT. 0.0) THEN
              ISPG = TENSTF(L,3+JKL,J)
              ISP = 9*(ISPG-1) + 1
              DOT(JKL) = TENSTF(L,30+ISP,J)*CRACK(L,137+NCK,J) + TENSTF(
     1            L,31+ISP,J)*CRACK(L,138+NCK,J) + TENSTF(L,32+ISP,J)*
     2            CRACK(L,139+NCK,J)
          ENDIF
  110 CONTINUE
      COSTH = DMAX1(DMAX1(DOT(1),DOT(2)),DOT(3))
      IF (TENSTF(L,4,J).GT.0 .OR. TENSTF(L,5,J).GT.0 .OR. TENSTF(L,6,J)
     1    .GT.0) THEN
          IF (DABS(COSTH) .GE. 0.866) THEN        !CHECK
              COFET = 1.0
          ELSE
              COFET = 10.0
          ENDIF
      ELSE
          COFET = 1.0
      ENDIF
      GF = GF*COFET*COSTH
```

```
      GF = DMAX1(PROPER(11),GF)
C
C               EPSNOT = STRAIN FOR ZERO STRESS ACROSS THE CRACK.
C
      IF (TENSTF(L,4,J).EQ.0 .AND. TENSTF(L,5,J).EQ.0 .AND. TENSTF(L,6,J
     1    ).EQ.0) THEN
          EPSNOT = (18.*GF)/(5.*SIGM1F*CL)
      ELSE
          EPSNOT = GF/(SIGM1F*CL*0.9)
      ENDIF
C
C                SAVE THE ULTIMATE CRACK STRAIN
C
      CRACK(L,127+NCRACK,J) = EPSNOT
C
C
C     GENERATE INITIAL DIRECT STIFFNESS ACROSS THE CRACK.THIS
C     MUST BE EQUAL TO INFINITY USING SECANT FORMULATION.ASSU-
C     MING THAT THE STRESS ACROSS THE CRACK REDUCES TO 99% OF
C     THE INITIATION STRESS DETERMINE THE INITIAL CRACK STIFF.
C
C
  120 CONTINUE
      SIGCRK = .90*SIGM1F
C
      CALL INITES (SIGCRK, PROPER, ES, NCRACK, L, J, IOUT)
      CRACK(L,87+NCRACK,J) = ES
C
C                INITIALIZE THE OLD AND NEW CRACK STRAINS.
C
      CRACK(L,47+NCRACK,J) = .10E-10
      CRACK(L,57+NCRACK,J) = .10E-10
C
C
C     GENERATE SHEAR STIFFNESS ALONG THE CRACK.FOR THIS
C     NEED THE SECANT VALUES OF 'E' & 'NU' OF CONCRETE AT
C     THE INITIATION OF CRACKING AND THE SHEAR RETENTION
C     FACTOR 'BETHA'.THE SHEAR STIFFNESS OF THE CRACK IS
C     ASSUMED TO REMAIN CONSTANT FOR THE ENTIRE CRACK HIST.
C     NOTE== THIS HAS BEEN MODIFIED TO A VARING BETA MODEL BASED
C     ON EXPERIMENTAL DATA OF VINTZELEOU AND TASSIOS.
C
      ESFLR = HISTO(L,26,J)
      NUSFLR = HISTO(L,27,J)
      GSFLR = ESFLR/(2.*(1.+NUSFLR))
      BETHA = PROPER(10)
C
C  CRACK SHEAR STIFFNESS FOR EXPERIMENT(VINTZELEOU)BASED SOFTENING
C
      GCRAK1 = GSFLR*10
      GCRAK2 = GSFLR*10
C
C  CRACK SHEAR STIFFNESS FOR PLAIN CONCRETE
C
      IF (TENSTF(L,4,J).EQ.0 .AND. TENSTF(L,5,J).EQ.0 .AND. TENSTF(L,6,J
     1    ).EQ.0) THEN
          BETHA = 0.10
          GCRAK1 = (BETHA*GSFLR)/(1.-BETHA)
          GCRAK2 = (BETHA*GSFLR)/(1.-BETHA)
      ENDIF
C
      CRACK(L,97+NCRACK,J) = GCRAK1
      CRACK(L,107+NCRACK,J) = GCRAK2
```

```
C              WRITE(*,*) 'GCRK1',GCRAK1,'GCRK2',GCRAK2,'BETHA',BETHA
C
C               FOR THE FIRST CRACK  CALCULATE AND STORE THE
C               PRINCIPAL STRAINS IN INTACT CONCRETE.
C
      SIGM2F = HISTO(L,35,J)
      SIGM3F = HISTO(L,36,J)
C
C              RETRIEVE THE CONSTITUTIVE MATRIX (IN PRINCIPAL AXES)
C
      ES = ESFLR
      NUS = NUSFLR
      CONST = ES/((1.-2.0*NUS)*(1+NUS))
      DEP(1,1) = CONST*(1.0-NUS)
      DEP(2,2) = CONST*(1.0-NUS)
      DEP(3,3) = CONST*(1.0-NUS)
      DEP(1,2) = CONST*NUS
      DEP(2,1) = DEP(1,2)
      DEP(3,1) = DEP(1,2)
      DEP(1,3) = DEP(1,2)
      DEP(2,3) = DEP(1,2)
      DEP(3,2) = DEP(1,2)
      DEP(4,4) = ES/(2.0*(1+NUS))
      DEP(5,5) = DEP(4,4)
      DEP(6,6) = DEP(4,4)
C              INVERT THE 'DEP' MATRIX.
C
C              N = NO. OF ROWS IN 'DEP'
C            NRC = MAX. NO. OF ROWS(AND COLUMNS) ALLOWED.
C            EPS = MIN. ALLOWABLE MAGNITUDE FOR A PIVOT ELEMENT.
C
C
      NR = 6
      NRC = 6
      EPS = 10.0E-20
C
C              INVERSE OF 'DEP' WILL BE RETURNED IN 'DEP'.
C
C              DETER = DETERMINANT OF 'DEP' THAT IS RETURNED AS THE
C                     VALUE OF FUNCTION 'SIMUL'.
C
C
      DETER = SIMUL(NR,DEP,EPS,NRC,IOUT)
      IF (DETER .EQ. 0.0) THEN
         WRITE (IOUT, 900) L, J, IELEM
         STOP
      ENDIF
C
      EPS1 = DEP(1,1)*SIGM1F + DEP(1,2)*SIGM2F + DEP(1,3)*SIGM3F
      EPS2 = DEP(2,1)*SIGM1F + DEP(2,2)*SIGM2F + DEP(2,3)*SIGM3F
      EPS3 = DEP(3,1)*SIGM1F + DEP(3,2)*SIGM2F + DEP(3,3)*SIGM3F
      AEMAX = DMAX1(DMAX1(DABS(EPS1),DABS(EPS2)),DABS(EPS3))
      IF (DABS(EPS1) .LE. 1E-3*AEMAX) EPS1 = 0.0
      IF (DABS(EPS2) .LE. 1E-3*AEMAX) EPS2 = 0.0
      IF (DABS(EPS3) .LE. 1E-3*AEMAX) EPS3 = 0.0
      IF (NCRACK .EQ. 1) HISTO(L,37,J) = DMAX1(DMAX1(EPS1,EPS2),EPS3)
C
C     TENSION STIFFENING IN STEEL DIRECTION
C
      IFLAG1 = 0
      IFLAG2 = 0
      IFLAG3 = 0
      IFLAG4 = 0
```

```
      IFLAG5 = 0
      IFLAG6 = 0
      IMARK1 = 0
      IMARK2 = 0
      IMARK3 = 0
      MCHECK = 0
      DO 180 ISP = 1, 3
C
C CHECK THE DIRECTION OF THE CRACK WITH AN EXISTING TENSION-STIFFENING
C DIRECTION FOR(SECONDARY CRACKS).IF DIRECTION OF THIS CRACK NORMAL IS
C CLOSE TO THAT OF THE STEEL THEN NO STRAIN SOFTENING CONSIDERED. ELSE
C THIS IS A STRAIN SOFTENING CRACK AND MUST BE TREATED AS SUCH.
C
          IF (TENSTF(L,ISP,J) .EQ. 1.0) THEN
              IF (ISP .EQ. 1) IFLAG1 = 1
              IF (ISP .EQ. 2) IFLAG2 = 1
              IF (ISP .EQ. 3) IFLAG3 = 1
              ISPG = INT(TENSTF(L,3+ISP,J))
              ISMG = (ISPG-1)*9 + 1
C
C ... TENSION STIFFENING (STEEL) DIRECTION W.R.T TO 'GLOBAL'
C
              N(1,1) = TENSTF(L,ISMG+30,J)
              N(1,2) = TENSTF(L,ISMG+31,J)
              N(1,3) = TENSTF(L,ISMG+32,J)
              N(2,1) = TENSTF(L,ISMG+33,J)
              N(2,2) = TENSTF(L,ISMG+34,J)
              N(2,3) = TENSTF(L,ISMG+35,J)
              N(3,1) = TENSTF(L,ISMG+36,J)
              N(3,2) = TENSTF(L,ISMG+37,J)
              N(3,3) = TENSTF(L,ISMG+38,J)
C
C ... CRACK NORMAL W.R.T TO 'GLOBAL'
C
              NT(1,1) = HISTO(L,10,J)
              NT(2,1) = HISTO(L,11,J)
              NT(3,1) = HISTO(L,12,J)
              NT(1,2) = HISTO(L,13,J)
              NT(2,2) = HISTO(L,14,J)
              NT(3,2) = HISTO(L,15,J)
              NT(1,3) = HISTO(L,16,J)
              NT(2,3) = HISTO(L,17,J)
              NT(3,3) = HISTO(L,18,J)
C
              RDOT = N(1,1)*NT(1,1) + N(1,2)*NT(2,1) + N(1,3)*NT(3,1)
              IF (DABS(RDOT) .LT. 0.906) MCHECK = 1
              GO TO 170
          ENDIF
C
C SEE IF TENSION STIFFENING IS ACTIVATED FOR THE CRACK.
C
          IF (TENSTF(L,ISP,J) .EQ. 0.0) THEN
              ISPG = INT(TENSTF(L,3+ISP,J))
              IF (ISPG .EQ. 0) THEN
                  IF (ISP .EQ. 1) IFLAG1 = 1
                  IF (ISP .EQ. 2) IFLAG2 = 1
                  IF (ISP .EQ. 3) IFLAG3 = 1
                  GO TO 170
              ENDIF
              IF (ISPG .GT. 0) THEN
                  ISMG = (ISPG-1)*9 + 1
C
C ... TENSION STIFFENING (STEEL) DIRECTION W.R.T TO 'GLOBAL'
```

```
C
                N(1,1) = TENSTF(L,ISMG+30,J)
                N(1,2) = TENSTF(L,ISMG+31,J)
                N(1,3) = TENSTF(L,ISMG+32,J)
                N(2,1) = TENSTF(L,ISMG+33,J)
                N(2,2) = TENSTF(L,ISMG+34,J)
                N(2,3) = TENSTF(L,ISMG+35,J)
                N(3,1) = TENSTF(L,ISMG+36,J)
                N(3,2) = TENSTF(L,ISMG+37,J)
                N(3,3) = TENSTF(L,ISMG+38,J)
C
C ... CRACK NORMAL W.R.T TO 'GLOBAL'
C
                NT(1,1) = HISTO(L,10,J)
                NT(2,1) = HISTO(L,11,J)
                NT(3,1) = HISTO(L,12,J)
                NT(1,2) = HISTO(L,13,J)
                NT(2,2) = HISTO(L,14,J)
                NT(3,2) = HISTO(L,15,J)
                NT(1,3) = HISTO(L,16,J)
                NT(2,3) = HISTO(L,17,J)
                NT(3,3) = HISTO(L,18,J)


C
                RDOT=N(1,1)*NT(1,1)+N(1,2)*NT(2,1)+N(1,3)*NT(3,1)
C
C
C ... FIND THE ORIENTATION OF THE CRACK WITH RESPECT TO STEEL AND
C     INITITATE A TEN. STIF SPRING IF NECESSARY.
C
                DO 140 IMM = 1, 3
                    DO 130 IMP = 1, 3
                        ROT(IMM,IMP) = 0.0
                        ROT(IMM,IMP) = ROT(IMM,IMP) + N(IMM,1)*NT(1,
    1                       IMP)
                        ROT(IMM,IMP) = ROT(IMM,IMP) + N(IMM,2)*NT(2,
    1                       IMP)
                        ROT(IMM,IMP) = ROT(IMM,IMP) + N(IMM,3)*NT(3,
    1                       IMP)
  130               CONTINUE
  140           CONTINUE
C
                DO KK = 1, 3
                    DO KM = 1, 3
                        IF(DABS(ROT(KK,KM)).LE.0.01)ROT(KK,KM)=0.0
                    END DO
                END DO
C
C... STRAIN IN TENSIONSTIFFENING (STEEL) DIRECTION IS...
C
                EPSSTL = EPS1*ROT(1,1)*ROT(1,1) + EPS2*ROT(1,2)*ROT(1,
    1               2) + EPS3*ROT(1,3)*ROT(1,3)
C
                IF (DABS(EPSSTL) .LT. 10E-9) EPSSTL = 0.0
                IF (ISP .EQ. 1) IMARK1 = 1
                IF (ISP .EQ. 2) IMARK2 = 1
                IF (ISP .EQ. 3) IMARK3 = 1
                IF (DABS(RDOT).LE.0.25 .AND. EPSSTL.LT.0.0) THEN
                    IF (ISP .EQ. 1) IMARK1 = 0
                    IF (ISP .EQ. 2) IMARK2 = 0
                    IF (ISP .EQ. 3) IMARK3 = 0
                    IF (ISP .EQ. 1) IFLAG4 = 2
                    IF (ISP .EQ. 2) IFLAG5 = 2
```

```
                  IF (ISP .EQ. 3) IFLAG6 = 2
               ENDIF
C
               IF (DABS(RDOT) .GE. 0.25) THEN
C... ACTIVATE TENSIONSTIFFENING SPRINGS WHEN STRAINS IN THE SPRING DIR.
C    AT CRACKING EXCEED UNIAXIAL CRACKING STRAINS.
C
                  CKSTIN = PROPER(6)/PROPER(1)
                  IF (EPSSTL .GE. CKSTIN) THEN
                     MCHECK = 0
                     IF (ISP .EQ. 1) IFLAG1 = 1
                     IF (ISP .EQ. 2) IFLAG2 = 1
                     IF (ISP .EQ. 3) IFLAG3 = 1
                     TENSTF(L,ISP,J) = 1.0
                  ELSE IF (EPSSTL.GE.0.0 .AND. EPSSTL.LT.CKSTIN)
     1                  THEN
                     IF (ISP .EQ. 1) IFLAG4 = 2
                     IF (ISP .EQ. 2) IFLAG5 = 2
                     IF (ISP .EQ. 3) IFLAG6 = 2
                  ENDIF
C
C STORE THE CRACKING STRAIN AT UNIAXIAL CRACKING STRAINS USED AS A
C PARAMETER.
C
                  IF (CKSTIN .LE. 0.0) THEN
                     TENSTF(L,9+ISPG,J) = 0.0
                  ELSE
                     TENSTF(L,9+ISPG,J) = CKSTIN
                  ENDIF
C
                  AL1 = ROT(1,1)
                  AM1 = ROT(2,1)
                  AN1 = ROT(3,1)
                  AL2 = ROT(1,2)
                  AM2 = ROT(2,2)
                  AN2 = ROT(3,2)
                  AL3 = ROT(1,3)
                  AM3 = ROT(2,3)
                  AN3 = ROT(3,3)
C
C CHECK FOR SPECIAL CASES.....
C
                  IF (TENSTF(L,4,J).EQ.0 .OR. TENSTF(L,5,J).EQ.0
     1                .OR. TENSTF(L,6,J).EQ.0) THEN
                     IF (DABS(AN3) .LE. 0.01) THEN
                        AL1 = ROT(1,1)
                        AL2 = ROT(1,3)
                        AL3 = ROT(1,2)
                        AM1 = ROT(2,1)
                        AM2 = ROT(2,3)
                        AM3 = ROT(2,2)
                        AN1 = ROT(3,1)
                        AN2 = ROT(3,3)
                        AN3 = ROT(3,2)
                     ENDIF
                     IF (DABS(AN3) .LE. 0.01) WRITE (*, *) 'AN3',
     1                  AN3
                     IF (DABS(AL1) .LE. 0.01) WRITE (*, *) 'AL1',
     1                  AL1
                     IF (DABS(AM1) .LE. 0.01) WRITE (*, *) 'AM1',
     1                  AM1
                  ENDIF
C
```

```
                   IF (TENSTF(L,4,J).EQ.0 .AND. TENSTF(L,5,J).EQ.0
     1                 .OR. TENSTF(L,5,J).EQ.0 .AND. TENSTF(L,6,J)
     2                .EQ.0 .OR. TENSTF(L,6,J).EQ.0 .AND. TENSTF(L,4
     3                ,J).EQ.0) THEN
                      IF (DABS(AM2) .LE. 0.01) THEN
                         AL1 = ROT(1,1)
                         AL2 = ROT(1,3)
                         AL3 = ROT(1,2)
                         AM1 = ROT(2,1)
                         AM2 = ROT(2,3)
                         AM3 = ROT(2,2)
                         AN1 = ROT(3,1)
                         AN2 = ROT(3,3)
                         AN3 = ROT(3,2)
                      ENDIF
                      IF (DABS(AM2) .LE. 0.01) WRITE (*, *) 'AM2',
     1                   AM2, 'AL1', AL1
                   ENDIF
C EVALUATE THE SPRING INITAITING STRESSES AND STORE THEM HERE IN CASE
C IT IS DETERMINED TO BE ACTIVATED ALONG WITH ANOTHER SPRING DIRECTION
C IN THIS STEP.
C
                   IF (TENSTF(L,4,J).GT.0.0 .AND. TENSTF(L,5,J).GT.
     1                0.0 .AND. TENSTF(L,6,J).GT.0.0) THEN
C
C ... ALL THREE STEEL DIRECTIONS PRESENT (NEED NOT BE ACTIVE).
C
                      TENSTF(L,6+ISPG,J) = SIGM1F
C                     WRITE(*,*)'THREE SIGF',SIGF,'SIGM1F',SIGM1F
C
                   ELSE IF (TENSTF(L,4,J).GT.0.0 .AND. TENSTF(L,5,J)
     1                   .GT.0.0 .AND. TENSTF(L,6,J).EQ.0.0 .OR.
     2                   TENSTF(L,5,J).GT.0.0 .AND. TENSTF(L,6,J)
     3                   .GT.0.0 .AND. TENSTF(L,4,J).EQ.0.0 .OR.
     4                   TENSTF(L,4,J).GT.0.0 .AND. TENSTF(L,6,J)
     5                   .GT.0.0 .AND. TENSTF(L,5,J).EQ.0.0) THEN
C
C ... ANY TWO DIRECTIONS PRESENT (NEED NOT BE ACTIVE).
C
                      SIGF = SIGM1F*((AL3*AN1-AL1*AN3)*(AM2*AN3-AM3*
     1                   AN2)-(AN1*AM3-AM1*AN3)*(AL2*AN3-AN2*AL3))/
     2                   (AN3*(AM1*(AL2*AN3-AN2*AL3)-AL1*(AM2*AN3-
     3                   AM3*AN2)))
C
C                     WRITE(*,*)'ANY TWO SIGF',SIGF,'SIGM1F',SIGM1F
                      TENSTF(L,6+ISPG,J) = SIGF
C
                   ELSE IF (TENSTF(L,4,J).GT.0.0 .AND. TENSTF(L,5,J)
     1                   .EQ.0.0 .AND. TENSTF(L,6,J).EQ.0.0 .OR.
     2                   TENSTF(L,5,J).GT.0.0 .AND. TENSTF(L,6,J)
     3                   .EQ.0.0 .AND. TENSTF(L,4,J).EQ.0.0 .OR.
     4                   TENSTF(L,4,J).GT.0.0 .AND. TENSTF(L,6,J)
     5                   .EQ.0.0 .AND. TENSTF(L,5,J).EQ.0.0) THEN
C
                      SIGF = SIGM1F*((AN1*AM2-AM1*AN2)*(AL3*AM2-AM3*
     1                   AL2)-(AL2*AM1-AM2*AL1)*(AM3*AN2-AN3*AM2))/
     2                   (AL1*AM2*(AM3*AN2-AN3*AM2))
C
C
C ...ONLY ONE DIRECTION PRESENT (NEED NOT BE ACTIVE).
C
C                     WRITE(*,*)'ONE ONLY SIGF',SIGF,'SIGM1F',SIGM1F
                      TENSTF(L,6+ISPG,J) = SIGF
```

```
C
                        ENDIF
C
C SET THE SPRING INITAITING STRAINS HERE FOR CONTINOUS MONITORING DURING
C THE LOADING PROCEDURE.
C
                        IF (EPSSTL .LE. 0.0) THEN
                            TENSTF(L,15+ISPG,J) = 0.0
                            TENSTF(L,18+ISPG,J) = 0.0
                        ELSE IF (EPSSTL .GE. CKSTIN) THEN
                            TENSTF(L,15+ISPG,J) = CKSTIN
                            TENSTF(L,18+ISPG,J) = CKSTIN
                        ELSE
                            TENSTF(L,15+ISPG,J) = EPSSTL
                            TENSTF(L,18+ISPG,J) = EPSSTL
                        ENDIF
C
C SET THE SPRING INITAITING STIFFNESS HERE FOR POSSIBLE ACTIVATION
C DURING THE LOADING PROCEDURE
C
                        IF (EPSSTL .GE. CKSTIN) THEN
                            TENSTF(L,ISPG+24,J) = SIGF/CKSTIN
                        ELSE IF (EPSSTL.LT.CKSTIN .AND. EPSSTL.GT.0.0)
     1                          THEN
                            TENSTF(L,ISPG+24,J) = SIGF/EPSSTL
                        ENDIF
C
C SET THE SPRING INITIATING STATE TO LOADING
C
                        TENSTF(L,21+ISPG,J) = 1.0
                    ENDIF
                ENDIF
            ENDIF
  170       CONTINUE
  180 CONTINUE
C
C ... IF THE CURRENT CRACK IS NOT STRAIN SOFTENING THEN
C     CHECK IF A TENSION STIFFENING DIRECTION IS ACTIVATED AT THIS
C     STAGE ELSE CONTINUE STRAIN SOFTENING UNTIL THE STRAINS PICK UP
C     FOR THE ACTIVATION OF TENSION STIFFENING
C
C       WRITE(*,*) 'CKINIT FLAG CHK',CRACK(L,87+NCRACK,J)
C       WRITE(*,*)'IFG1',IFLAG1,'IFG2',IFLAG2,'IFG3',IFLAG3
C       WRITE(*,*)'IFG4',IFLAG4,'IFG5',IFLAG5,'IFG6',IFLAG6
      IF (MCHECK .EQ. 0) THEN
          IF (IFLAG1.EQ.1 .AND. IFLAG2.EQ.1 .AND. IFLAG3.EQ.1) THEN
              ITCRK = NCRACK
          ELSE IF (IFLAG1.EQ.1 .AND. IFLAG2.EQ.1 .AND. IFLAG3.EQ.0 .OR.
     1            IFLAG1.EQ.1 .AND. IFLAG3.EQ.1 .AND. IFLAG2.EQ.0 .OR.
     2            IFLAG2.EQ.1 .AND. IFLAG3.EQ.1 .AND. IFLAG1.EQ.0) THEN
              IF (IFLAG4.EQ.2 .OR. IFLAG5.EQ.2 .OR. IFLAG6.EQ.2) THEN
                  IF (IFLAG1.EQ.1 .AND. TENSTF(L,1,J).EQ.1 .OR. IFLAG2
     1                .EQ.1 .AND. TENSTF(L,2,J).EQ.1 .OR. IFLAG3.EQ.1
     2                 .AND. TENSTF(L,3,J).EQ.1) THEN
                      ITCRK = NCRACK
                      IF (IFLAG4.EQ.2 .AND. IMARK1.EQ.1) TENSTF(L,1,J)
     1                     = 1.0
                      IF (IFLAG5.EQ.2 .AND. IMARK2.EQ.1) TENSTF(L,2,J)
     1                     = 1.0
                      IF (IFLAG6.EQ.2 .AND. IMARK3.EQ.1) TENSTF(L,3,J)
     1                     = 1.0
                      IFLAG1 = 1
                      IFLAG2 = 1
```

```
                  IFLAG3 = 1
               ENDIF
            ELSE IF (TENSTF(L,4,J).EQ.0.0 .OR. TENSTF(L,5,J).EQ.0.0
     1                .OR. TENSTF(L,6,J).EQ.0.0) THEN
               ITCRK = NCRACK
               IFLAG1 = 1
               IFLAG2 = 1
               IFLAG3 = 1
            ENDIF
         ELSE IF (IFLAG1.EQ.1 .AND. IFLAG2.EQ.0 .AND. IFLAG3.EQ.0 .OR.
     1         IFLAG2.EQ.1 .AND. IFLAG3.EQ.0 .AND. IFLAG1.EQ.0 .OR.
     2         IFLAG3.EQ.1 .AND. IFLAG1.EQ.0 .AND. IFLAG2.EQ.0) THEN
            IF (IFLAG4.EQ.2 .AND. IFLAG5.EQ.2 .AND. IFLAG6.EQ.0 .OR.
     1         IFLAG5.EQ.2 .AND. IFLAG6.EQ.2 .AND. IFLAG4.EQ.0 .OR.
     2         IFLAG6.EQ.2 .AND. IFLAG4.EQ.2 .AND. IFLAG5.EQ.0) THEN
               IF (IFLAG3.EQ.1 .AND. TENSTF(L,6,J).GT.0.0 .OR. IFLAG2
     1             .EQ.1 .AND. TENSTF(L,5,J).GT.0.0 .OR. IFLAG1.EQ.1
     2              .AND. TENSTF(L,4,J).GT.0.0) THEN
                  IF (IFLAG4 .EQ. 2) TENSTF(L,1,J) = 1.0
                  IF (IFLAG5 .EQ. 2) TENSTF(L,2,J) = 1.0
                  IF (IFLAG6 .EQ. 2) TENSTF(L,3,J) = 1.0
                  ITCRK = NCRACK
                  IFLAG1 = 1
                  IFLAG2 = 1
                  IFLAG3 = 1
               ENDIF
            ELSE IF (TENSTF(L,4,J).EQ.1.0 .AND. TENSTF(L,5,J).EQ.0.0
     1              .AND. TENSTF(L,6,J).EQ.0.0 .OR. TENSTF(L,5,J).EQ.
     2             1.0 .AND. TENSTF(L,4,J).EQ.0.0 .AND. TENSTF(L,6,J)
     3             .EQ.0.0 .OR. TENSTF(L,6,J).EQ.1.0 .AND. TENSTF(L,5
     4             ,J).EQ.0.0 .AND. TENSTF(L,4,J).EQ.0.0) THEN
               ITCRK = NCRACK
               IFLAG1 = 1
               IFLAG2 = 1
               IFLAG3 = 1
            ENDIF
         ENDIF
      ENDIF
C       IF(ELNUM.EQ.18) THEN
C       WRITE(*,*) 'CKINIT EXIT STIF',CRACK(L,87+NCRACK,J)
C       END IF
      IF (IFLAG1.EQ.1 .AND. IFLAG2.EQ.1 .AND. IFLAG3.EQ.1) THEN
C       IF(ELNUM.EQ.18) THEN
C         WRITE(*,*) 'ALL THREE FLAGS 1 IN CKINIT'
C       END IF
         DO 200 IM = 1, 6
            DO 190 IN = 1, 6
               DTSTIF(IM,IN) = 0.0
  190       CONTINUE
  200    CONTINUE
         DO 250 ISP = 1, 3
            ISPG = TENSTF(L,3+ISP,J)
            IF (ISPG.GT.0 .AND. TENSTF(L,ISP,J).EQ.1.0) THEN
               ISMG = (ISPG-1)*9 + 1
C
C ... TENSION STIFFENING (STEEL) DIRECTION W.R.T TO 'GLOBAL'
C
               N(1,1) = TENSTF(L,ISMG+30,J)
               N(1,2) = TENSTF(L,ISMG+31,J)
               N(1,3) = TENSTF(L,ISMG+32,J)
               N(2,1) = TENSTF(L,ISMG+33,J)
               N(2,2) = TENSTF(L,ISMG+34,J)
               N(2,3) = TENSTF(L,ISMG+35,J)
```

```
                N(3,1) = TENSTF(L,ISMG+36,J)
                N(3,2) = TENSTF(L,ISMG+37,J)
                N(3,3) = TENSTF(L,ISMG+38,J)
                DO 220 IM = 1, 6
                    DO 210 IN = 1, 6
                        TENXYZ(IM,IN) = 0.0
  210               CONTINUE
  220           CONTINUE
C                      WRITE(*,*)'CKINIT TSTIF',TENSTF(L,24+ISPG,J)
                TENXYZ(1,1) = TENSTF(L,24+ISPG,J)
                CALL DTRANS (TENXYZ, N, IOUT)
                DO 240 IJ = 1, 6
                    DO 230 JK = 1, 6
                        DTSTIF(IJ,JK) = TENXYZ(IJ,JK) + DTSTIF(IJ,JK)
C                      WRITE(*,*)'IJ',IJ,'JK',JK,'STIF',DTSTIF(IJ,JK)
  230               CONTINUE
  240           CONTINUE
            ENDIF
  250     CONTINUE
      ENDIF
      RETURN
  900    FORMAT(//,1X,'    ERROR',/,5X,'NEGATIVE PIVOT ENCOUNTERED'
     1          ,' DURING INVERSION OF CONSTITUTIVE MATRIX FOR INTACT '
     2          ,'CONCRETE ...CKINIT  STOP 3','G.PT',I2,'LAYER',I2,
     3          'ELNUM',I2)
      END
C
C
C***************************************************************
C     INCLUDE(PROCESS)
C
      SUBROUTINE CRKPNT(EPSX,EPSY,EPSZ,GAMAXY,GAMAYZ,GAMAXZ,PROPER,
     $ TOLERE,CHKGUS,NSTIFF,CONVER,IOUT,EPS1,EPS2,EPS3,ELNUM,MAXCRK,
     $ MAXCK2,SIGMA1,SIGMA2,SIGMA3,L,J,PDIR,NLAYRS,NNEL,ITYPE,IFLAG8,
     $ ISOFT,ITCRK)
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C DMATCS    PSTSTN    VINTAS    ARANGE    CRACKE    CRKCON
C OVERFT    SIMUL
C
C************************************************************
      INTEGER ELNUM
      REAL*8 LTHICK
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON/ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      COMMON/MATER1/DEP(6,6)
      COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),MLAYRS,MATRL,LYNUM
      DIMENSION PROPER(25),SIGMA(6),DCON(30,30)
      REAL*8 NUS,NT(3,6),PVAL(3),PDIR(3,3),DN(3,3),DNT(3,3)
      LOGICAL CHKGUS
C
C
C      SUBROUTINE TO CHECK STATUS AT A CRACKED POINT AND
C      FINALLY RETURN THE PRINCIPAL STRAINS OF INTACT CON-
C      CRETE AND TOTAL STRESSES FOR FURTHER INVESTIGATION.
C
```

```
C
C     ......ASSUME A SINGLE CRACK AND INTACT CONCRETE CONNECTED
C           IN SERIES AND UNDER UNIAXIAL TENSION. THE EQUIVALENT
C           STIFFNESS OF THIS SYSTEM IS VERY CLOSE TO THE STIFFN-
C           ESS OF THE INTACT CONCRETE FOR CRACK STIFFNESSES MANY
C           TIMES LARGER THAN THE INTACT CONCRETE MODULUS OF ELST.
C           THEREFORE, THE SELECTED TOLERANCE WILL PRACTICALLY BE
C           EFFECTIVE FOR CRACK STIFFNESSES SMALLER THAN THE INTACT
C           CONCRETE STIFFNESS.
C
C         ......  TOLERE = TOLERANCE USED FOR INTACT CONCRETE
C
C
      TOLER = TOLERE
      ISOFT = 0
      ITCRK = 0
C
C         .... RETRIEVE THE GLOBAL COMPOSITE CONSTITUTIVE MATRIX.
C
      ICODE = 1
      IPG = 0
      CALL DMATCS (ELNUM, ITYPE, L, IFLAG8, IOUT, ICODE, IPG)
      D11 = DEP(1,1)
      D12 = DEP(1,2)
      D13 = DEP(1,3)
      D14 = DEP(1,4)
      D15 = DEP(1,5)
      D16 = DEP(1,6)
      D21 = DEP(2,1)
      D22 = DEP(2,2)
      D23 = DEP(2,3)
      D24 = DEP(2,4)
      D25 = DEP(2,5)
      D26 = DEP(2,6)
      D31 = DEP(3,1)
      D32 = DEP(3,2)
      D33 = DEP(3,3)
      D34 = DEP(3,4)
      D35 = DEP(3,5)
      D36 = DEP(3,6)
      D41 = DEP(4,1)
      D42 = DEP(4,2)
      D43 = DEP(4,3)

      D44 = DEP(4,4)
      D45 = DEP(4,5)
      D46 = DEP(4,6)
      D51 = DEP(5,1)
      D52 = DEP(5,2)
      D53 = DEP(5,3)
      D54 = DEP(5,4)
      D55 = DEP(5,5)
      D56 = DEP(5,6)
      D61 = DEP(6,1)
      D62 = DEP(6,2)
      D63 = DEP(6,3)
      D64 = DEP(6,4)
      D65 = DEP(6,5)
      D66 = DEP(6,6)
C
C     HAVING THE GLOBAL STRAINS DETERMINE GLOBAL STRESSES.
C
C     ...... REDUCE TRUNCATION ERRORS DURING ADDITION/SUBT.
```

```
C
      XYZ = DMAX1(DMAX1(DMAX1(DMAX1(DMAX1(DABS(EPSX),DABS(EPSY)),DABS(
     1    GAMAXY)),DABS(GAMAYZ)),DABS(GAMAXZ)),DABS(EPSZ))
      IF (DABS(EPSX) .LT. XYZ*10.0E-7) EPSX = 0.0
      IF (DABS(EPSY) .LT. XYZ*10.0E-7) EPSY = 0.0
      IF (DABS(EPSZ) .LT. XYZ*10.0E-7) EPSZ = 0.0
      IF (DABS(GAMAXY) .LT. XYZ*10.0E-7) GAMAXY = 0.0
      IF (DABS(GAMAYZ) .LT. XYZ*10.0E-7) GAMAYZ = 0.0
      IF (DABS(GAMAXZ) .LT. XYZ*10.0E-7) GAMAXZ = 0.0
      SIGMA(1) = D11*EPSX + D12*EPSY + D14*GAMAXY + D15*GAMAYZ + D16*
     1    GAMAXZ + D13*EPSZ
      SIGMA(2) = D21*EPSX + D22*EPSY + D24*GAMAXY + D25*GAMAYZ + D26*
     1    GAMAXZ + D23*EPSZ
      SIGMA(3) = D31*EPSX + D32*EPSY + D34*GAMAXY + D35*GAMAYZ + D36*
     1    GAMAXZ + D33*EPSZ
      SIGMA(4) = D41*EPSX + D42*EPSY + D44*GAMAXY + D45*GAMAYZ + D46*
     1    GAMAXZ + D43*EPSZ
      SIGMA(5) = D51*EPSX + D52*EPSY + D54*GAMAXY + D55*GAMAYZ + D56*
     1    GAMAXZ + D53*EPSZ
      SIGMA(6) = D61*EPSX + D62*EPSY + D64*GAMAXY + D65*GAMAYZ + D66*
     1    GAMAXZ + D63*EPSZ
      XYZ = DMAX1(DMAX1(DMAX1(DMAX1(DMAX1(DABS(SIGMA(1)),DABS(SIGMA(2)))
     1    ,DABS(SIGMA(3))),DABS(SIGMA(4))),DABS(SIGMA(5))),DABS(SIGMA(6)
     2    ))
      IF (DABS(SIGMA(1)) .LT. XYZ*10.0E-4) SIGMA(1) = 0.0
      IF (DABS(SIGMA(2)) .LT. XYZ*10.0E-4) SIGMA(2) = 0.0
      IF (DABS(SIGMA(3)) .LT. XYZ*10.0E-4) SIGMA(3) = 0.0
      IF (DABS(SIGMA(4)) .LT. XYZ*10.0E-4) SIGMA(4) = 0.0
      IF (DABS(SIGMA(5)) .LT. XYZ*10.0E-4) SIGMA(5) = 0.0
      IF (DABS(SIGMA(6)) .LT. XYZ*10.0E-4) SIGMA(6) = 0.0
      IFG = 2
C
C              DETERMINE NUMBER OF CRACKS AT THIS POINT.
C
      NCRACK = INT(CRACK(L,1,J))
      DO 150 ICRACK = 1, NCRACK
C
C
C              DETERMINE TRANSPOSE OF THE TRANSFORMATION MATRIX 'N'.
C
         ICK = 9*(ICRACK-1) + 1
         AL1 = CRACK(L,137+ICK,J)
         AM1 = CRACK(L,138+ICK,J)
         AN1 = CRACK(L,139+ICK,J)
         AL2 = CRACK(L,140+ICK,J)
         AM2 = CRACK(L,141+ICK,J)
         AN2 = CRACK(L,142+ICK,J)
         AL3 = CRACK(L,143+ICK,J)
         AM3 = CRACK(L,144+ICK,J)
         AN3 = CRACK(L,145+ICK,J)
C
         NT(1,1) = AL1*AL1
         NT(1,2) = AM1*AM1
         NT(1,3) = AN1*AN1
         NT(1,4) = 2.0*AL1*AM1
         NT(1,5) = 2.0*AM1*AN1
         NT(1,6) = 2.0*AN1*AL1
         NT(2,1) = AL1*AL2
         NT(2,2) = AM1*AM2
         NT(2,3) = AN1*AN2
         NT(2,4) = AL1*AM2 + AM1*AL2
         NT(2,5) = AM1*AN2 + AN1*AM2
         NT(2,6) = AN1*AL2 + AL1*AN2
```

```
      NT(3,1) = AL1*AL3
      NT(3,2) = AM1*AM3
      NT(3,3) = AN1*AN3
      NT(3,4) = AM1*AL3 + AL1*AM3
      NT(3,5) = AN1*AM3 + AM1*AN3
      NT(3,6) = AL1*AN3 + AN1*AL3
C
C          CALCULATE CRACK INTERFACE STRESSES.
C
C          SIGCRK = DIRECT STRESS ACROSS THE CRACK.
C          TAUCRK = SHEAR    ;     ALONG   ;    ;  .
C
C
      SIGCRK = 0.0
      TAUCK1 = 0.0
      TAUCK2 = 0.0
      DO 110 II = 1, 3
          DO 100 JJ = 1, 6
              IF (II .EQ. 1) THEN
                  SIGCRK = SIGCRK + NT(II,JJ)*SIGMA(JJ)
              ELSE IF (II .EQ. 2) THEN
                  TAUCK1 = TAUCK1 + NT(II,JJ)*SIGMA(JJ)
              ELSE
                  TAUCK2 = TAUCK2 + NT(II,JJ)*SIGMA(JJ)
              ENDIF
  100     CONTINUE
  110 CONTINUE
C
      IF (DABS(TAUCK1) .LT. DABS(0.01*PROPER(6))) TAUCK1 = 0.0
      IF (DABS(TAUCK2) .LT. DABS(0.01*PROPER(6))) TAUCK2 = 0.0
C
      DCRACK = CRACK(L,87+ICRACK,J)
      GCRCK1 = CRACK(L,97+ICRACK,J)
      GCRCK2 = CRACK(L,107+ICRACK,J)
C
C          DETERMINE CRACK INTERFACE STRAINS.
C
C          NO COUPLING BETWEEN CRACK'S DIRECT AND SHEAR
C          STIFFNESSES ; THEREFORE,
C
C
      EPSCRK = SIGCRK/DCRACK
      GMCRK1 = TAUCK1/GCRCK1
      GMCRK2 = TAUCK2/GCRCK2
C
C -PRIN. STRESSES NEEDED FOR SHEAR STIF. CALCULATIONS.
      CALL PSTSTN (SIGMA, PVAL, PDIR, IFG, IOUT)
      SIGMA1 = PVAL(1)
      SIGMA1 = DMIN1(PROPER(6),SIGMA1)
      SIGMA2 = PVAL(2)
      SIGMA3 = PVAL(3)
      SMAX = DMAX1(DMAX1(DABS(SIGMA1),DABS(SIGMA2)),DABS(SIGMA3))
      IF (DABS(SIGMA1) .LE. SMAX*1.E-4) SIGMA1 = 0.0
      IF (DABS(SIGMA2) .LE. SMAX*1.E-4) SIGMA2 = 0.0
      IF (DABS(SIGMA3) .LE. SMAX*1.E-4) SIGMA3 = 0.0
      CALL VINTAS (GMCRK1, GMCRK2, EPSCRK, SIGMA1, SIGMA2, SIGMA3,
     1    PDIR, ICRACK, L, J, IOUT, ELNUM, NNEL, NLAYRS)
C
C          RETRIEVE THE CRACK'S PREVIOUS STRAIN FOR COMPARISON.
C
      EPSOLD = CRACK(L,47+ICRACK,J)
C
C           DETERMINE THE STATE OF THE CRACK AT THE END OF
```

```
C               THE PREVIOUS ITERATION.
C
          STATE = CRACK(L,117+ICRACK,J)
C
C  .... DUGDALE MODEL FOR REINFORCED CRACKED POINTS PRIOR TO THE
C       INITIATION OF TENSIONSTIFFENING.
C
          EPSULT = CRACK(L,127+ICRACK,J)
          AREA = CRACK(L,37+ICRACK,J)*EPSULT*0.9
          SUBARA = 0.5*CRACK(L,37+ICRACK,J)*(EPSCRK-EPSOLD)
C
C    CHECK FOR CONVERGENCE ON THE DUGDALE ENVELOPE.
C
          IF (TENSTF(L,4,J).GT.0.0 .OR. TENSTF(L,5,J).GT.0.0 .OR. TENSTF
     1        (L,6,J).GT.0.0) THEN
C
              IFLAG1 = 0
              IFLAG2 = 0
              IFLAG3 = 0
              DO 120 ISP = 1, 3
C
C CHECK THE DIRECTION OF THE CRACK WITH AN EXISTING STEEL
C DIRECTION FOR(SECONDARY CRACKS).IF DIRECTION OF THIS CRACK NORMAL IS
C CLOSE TO THAT OF THE STEEL THEN NO STRAIN SOFTENING CONSIDERED. ELSE
C THIS IS A STRAIN SOFTENING CRACK AND MUST BE TREATED AS SUCH.
C
                  IF (TENSTF(L,ISP+3,J) .NE. 0.0) THEN
                      ISPG = INT(TENSTF(L,3+ISP,J))
                      ISMG = (ISPG-1)*9 + 1
C
C ... TENSION STIFFENING (STEEL) DIRECTION W.R.T TO 'GLOBAL'
C
                      DN(1,1) = TENSTF(L,ISMG+30,J)
                      DN(1,2) = TENSTF(L,ISMG+31,J)
                      DN(1,3) = TENSTF(L,ISMG+32,J)
                      DN(2,1) = TENSTF(L,ISMG+33,J)
                      DN(2,2) = TENSTF(L,ISMG+34,J)
                      DN(2,3) = TENSTF(L,ISMG+35,J)
                      DN(3,1) = TENSTF(L,ISMG+36,J)
                      DN(3,2) = TENSTF(L,ISMG+37,J)
                      DN(3,3) = TENSTF(L,ISMG+38,J)
C
C ... CRACK NORMAL W.R.T TO 'GLOBAL'
C
                      ICK = 9*(ICRACK-1) + 1
                      DNT(1,1) = CRACK(L,137+ICK,J)
                      DNT(2,1) = CRACK(L,138+ICK,J)
                      DNT(3,1) = CRACK(L,139+ICK,J)
                      DNT(1,2) = CRACK(L,140+ICK,J)
                      DNT(2,2) = CRACK(L,141+ICK,J)
                      DNT(3,2) = CRACK(L,142+ICK,J)
                      DNT(1,3) = CRACK(L,143+ICK,J)
                      DNT(2,3) = CRACK(L,144+ICK,J)
                      DNT(3,3) = CRACK(L,145+ICK,J)
C
                      RDOT = DN(1,1)*DNT(1,1) + DN(1,2)*DNT(2,1) + DN(1,
     1                    3)*DNT(3,1)
                      PI = 4.0*DATAN(1.0D0)
                      IF (DABS(RDOT) .LT. DCOS((46.0*PI)/180.0)) THEN
                          IF (ISP .EQ. 1) IFLAG1 = 1
                          IF (ISP .EQ. 2) IFLAG2 = 1
                          IF (ISP .EQ. 3) IFLAG3 = 1
                      ENDIF
```

```
                ELSE IF (TENSTF(L,ISP+3,J) .EQ. 0) THEN
                    IF (ISP .EQ. 1) IFLAG1 = 1
                    IF (ISP .EQ. 2) IFLAG2 = 1
                    IF (ISP .EQ. 3) IFLAG3 = 1
                ENDIF
  120       CONTINUE
            IF (IFLAG1.EQ.1 .AND. IFLAG2.EQ.1 .AND. IFLAG3.EQ.1)
     1          GO TO 130
C
            IF (EPSCRK .GE. EPSULT) THEN
                IF (TENSTF(L,1,J).EQ.0.0 .AND. TENSTF(L,2,J).EQ.0.0
     1               .AND. TENSTF(L,3,J).EQ.0.0) THEN
                    WRITE (*, *)
     1                  'WARNING STRAIN-SOFTENING LIMIT FOR CRACK',
     2                  'FRACTURE ENERGY FULLY DESSIPATED', ICRACK,
     3                  'GAUSS PT', L, 'LAYER', J, 'ELEM', ELNUM
                    ITCRK = ICRACK
                    ISOFT = 1
                ELSE
                    CRACK(L,87+ICRACK,J) = PROPER(1)/1000.0
                ENDIF
            ELSE IF (EPSCRK.LT.0.0 .AND. NCRACK.GT.1.0) THEN
                CALL ARANGE (ICRACK, NCRACK, L, J)
            ENDIF
            CRACK(L,87+ICRACK,J) = DMAX1(PROPER(1)/1000.0,CRACK(L,87+
     1          ICRACK,J))
            IF (CRACK(L,87+ICRACK,J) .LE. PROPER(1)/1000.0) THEN
                CRACK(L,67+ICRACK,J) = GMCRK1
                CRACK(L,77+ICRACK,J) = GMCRK2
                CRACK(L,47+ICRACK,J) = EPSCRK
                CRACK(L,57+ICRACK,J) = EPSCRK
                GO TO 140
            ENDIF
            C1 = (SUBARA/AREA)*100.0
            IF (C1 .GE. TOLER) THEN
                CSTIFF = (CRACK(L,37+ICRACK,J)*(1.0-0.2*EPSCRK/EPSULT)
     1              )/EPSCRK
                IF (CSTIFF .LE. CRACK(L,87+ICRACK,J)) THEN
                    CRACK(L,87+ICRACK,J) = CSTIFF
                    CSTIFF = 0.0
                    CHKGUS = .TRUE.
                    NSTIFF = 1
                    CONVER = 999.0
                ENDIF
            ELSE
C
C   CHECK FOR OVERSHOOTING OF THE STRESS POINT.
C
                IF (SIGCRK .GT. CRACK(L,37+ICRACK,J)) THEN
                    EPSCRK = EPSOLD + TOLER*AREA/(0.5*CRACK(L,37+
     1                  ICRACK,J)*100.)
                    CSTIFF = (CRACK(L,37+ICRACK,J)*(1.0-0.2*EPSCRK/
     1                  EPSULT))/EPSCRK
                    IF (CSTIFF .LE. CRACK(L,87+ICRACK,J)) THEN
                        CRACK(L,87+ICRACK,J) = CSTIFF
                        CSTIFF = 0.0
                        CONVER = 999.0
                        NSTIFF = 1
                        CHKGUS = .TRUE.
                    ENDIF
                ENDIF
            ENDIF
            CRACK(L,87+ICRACK,J) = DMAX1(PROPER(1)/1000.0,CRACK(L,87+
```

```
     1            ICRACK,J))
              CRACK(L,67+ICRACK,J) = GMCRK1
              CRACK(L,77+ICRACK,J) = GMCRK2
              CRACK(L,47+ICRACK,J) = EPSCRK
              CRACK(L,57+ICRACK,J) = EPSCRK
              GO TO 140
          ENDIF
C
C     THE FOLLOWING IS FOR PURE STRAIN SOFTENING CRACKS ONLY.
C                           -------------------
C                           *                 *
C                           *  OPENING CRACK  *
C                           *                 *
C                           -------------------
  130     CONTINUE
          IF (EPSCRK.GT.0.0 .AND. STATE.EQ.4.) THEN
C
C
C
C                A CLOSED CRACK STARTS TO OPEN.USE 'EPSOLD' TO
C                DETERMINE A NEW CROSS-CRACK STIFFNESS.
C
C
C    CRACK REOPENS ALONG A DAMAGED SECANT PATH AT CLOSURE. THIS
C    APPROACH NOT ACTIVE CURRENTLY-I.E.CRACK ASSUMED TO OPEN A FRESH.
C
              CALL CRACKE (EPSOLD, PROPER, ES, ICRACK, L, J)
              EINITL = PROPER(1)
              ES = DMAX1(EINITL/1000.0,ES)
              CRACK(L,87+ICRACK,J) = ES
C
C                FOR THE CURRENT ITERATION ASSUME THAT THE CRACK OPENS
C                UP TO ZERO STRAIN.
C
              CRACK(L,57+ICRACK,J) = 0.0
C
C                SET THE STATE OF THE CRACK TO RELOADING.
C
              CRACK(L,117+ICRACK,J) = 3.
C
C
C                UPDATE THE CONSTITUTIVE LAW, UPDATE THE CORRESPONDING
C                ELEMENT STIFFNESS MATRIX AND CARRY OUT FURTHER ITERAT.
C
C
              CHKGUS = .TRUE.
              NSTIFF = 1
              CONVER = 999.
              GO TO 140
          ENDIF
          IF (EPSCRK.GT.0.0 .AND. EPSCRK.GE.EPSOLD) THEN
C
C
C                IF THE CRACK WAS UNLOADING (CLOSING) OR RELOADING-
C                (REOPENING) IN THE PREVIOUS ITERATION, THEN SET
C                THE STATE OF THE CRACK TO LOADING AGAIN.
C
C
              IF(STATE.EQ.2..OR.STATE.EQ.3.)CRACK(L,ICRACK+117,J)=1.0
C
C
C               THE CRACK IS STILL OPENING.SAVE THE CURRENT NORMAL
C               STRAIN AS THE CRACK'S OLD STRAIN.CRACK CLOSURE OR
```

```
C              UNLOADING IN SUBSEQUENT STEPS MAY START FROM THIS
C              STORED STRAIN.
C
C
            CRACK(L,47+ICRACK,J) = EPSCRK
C
C
C               SAVE THE CURRENT STRAIN FOR DISTINGUISHING BETWEEN
C               UNLOADING OR RELOADING IN SUBSEQUENT STEPS.
C
C
            CRACK(L,57+ICRACK,J) = EPSCRK
C
C              SAVE THE SHEAR STRAIN ALONG THE CRACK.
C
            CRACK(L,67+ICRACK,J) = GMCRK1
            CRACK(L,77+ICRACK,J) = GMCRK2
C
C               IF THE CRACK WAS COMPLETELY OPEN ...
C
            EINITL = PROPER(1)
            IF (DCRACK .LE. EINITL/1000.0) GO TO 140
C
C               DETERMINE THE SECANT 'E' CORRESPONDING TO THE
C               CURRENT CRACK STRAIN, EPSCRK.
C
            CALL CRACKE (EPSCRK, PROPER, ES, ICRACK, L, J)
            IF (ES .LE. EINITL/1000.0) THEN
C
C
C               THE CRACK HAS COMPLETELY OPENED LEADING TO ZERO STIFF.
C               TO BE ABLE TO CALCULATE THE CRACK STRAIN IN SUBSEQUENT
C               ITERATIONS ARBITRARILY ASSIGN A SMALL 'E' FOR IT.
C
C
                CRACK(L,87+ICRACK,J) = EINITL/1000.0
                CHKGUS = .TRUE.
                NSTIFF = 1
                CONVER = 999.
                WRITE (*, *) 'CRKPNT 4 999 ELNUM', ELNUM
                GO TO 140
            ENDIF
C
C                       CHECK FOR CONVERGENCE
C
            FT = CRACK(L,37+ICRACK,J)
C
C               IFLAG = 0 ... CRACK STIFFNESS HAS CONVERGED
C               IFLAG = 1 ...    '        '     SHOULD BE UPDATED.
C
            IFLAG = 0
            CALL CRKCON (PROPER, DCRACK, J, ICRACK, L, EPSCRK, ES,
     1          CHKGUS, NSTIFF, CONVER, TOLER, IOUT, EPSOLD, IFLAG)
            IF (CONVER .EQ. 999.) WRITE (*, *) 'CRKPNT 5 999 ELNUM',
     1          ELNUM
            IF (SIGCRK .LE. FT) GO TO 140
C
C               ...... CORRECTION FOR OVERSHOOTING
C
            IF (SIGCRK.GT.FT .AND. IFLAG.EQ.0) THEN
                CALL OVERFT (EPSOLD, TOLER, L, ICRACK, J, PROPER,
     1              DCRACK, CHKGUS, NSTIFF, CONVER, IOUT, SIGCRK, ES,
     2              ELNUM)
```

```
                IF (CONVER .EQ. 999.) WRITE (*, *)

     1              'CRKPNT 6 999 ELNUM', ELNUM
                GO TO 140
             ENDIF
          ENDIF
C
          IF (EPSCRK.GT.0.0 .AND. EPSCRK.LT.EPSOLD) THEN
C
C
C                 -------------------------------------------
C                 *                                         *
C                 *  PARTIAL CLOSING OR REOPENING OF THE CRACK *
C                 *                                         *
C                 -------------------------------------------
C
C
C           .....  GET THE PREVIOUSLY REGISTERED CRACK STRAIN.
C
             PRECRK = CRACK(L,47+ICRACK,J)
C
C
C               THE CRACK IS CLOSING.THE STATE OF THE CRACK SHOULD BE
C               EQUAL TO 2.
C
C
             IF (EPSCRK .LT. PRECRK) CRACK(L,117+ICRACK,J) = 2.0
C
C
C               THE CRACK IS RELOADING(OPENING).THE STATE OF THE CRACK
C               MUST BE EQUAL TO 3.
C
             IF (EPSCRK .GE. PRECRK) CRACK(L,117+ICRACK,J) = 3.
C
C               SAVE THE CURRENT CRACK STRAINS.
C
             CRACK(L,57+ICRACK,J) = EPSCRK
             CRACK(L,ICRACK+67,J) = GMCRK1
             CRACK(L,ICRACK+77,J) = GMCRK2
             GO TO 140
          ENDIF
C
C
C                 ----------------------------------
C                 *                                *
C                 *  COMPLETE CLOSURE OF THE CRACK  *
C                 *                                *
C                 ----------------------------------
C
C
C
          IF (EPSCRK .LE. 0.0) THEN
C
C  IF THE CRACK CLOSES(**) THE STIFFNESS OF THE ELEMENT REFLECTS A
C  DAMAGE PLANE IN THAT DIRECTION. IF THE CRACK REOPENS THEN IT STILL
C  GOES BACK TO SECANT RELOADING STIFFNESS AT CLOSURE(**) OF THE CRACK.
C  THE SHEAR DAMAGE AT THE INTERFACE IS ALLOWED TO CONTINUE AS BEFORE.
C        IF(NEWFG1.EQ.2) THEN
C           CRACK(L,227+I,J)=CRACK(L,87+I,J)
C           STATE=4.0
C           ESINTC=HISTO(L,20,J)
C           CRACK(L,87+I,J)=10.*ESINTC
C           GO TO 30
```

```
C        END IF
C
C
C              IF A VERY STIFF CRACK IS CLOSING ELIMINATE THIS
C              CRACK FROM THE 'CRACK' ARRAY AND DO NOT CONSIDER
C              IT IN THE FOLLOWING ITERATIONS.
C
C              FOR THE LAST CRACK ......
C
            EINITAL = PROPER(1)
C
C THIS IS CASE OF A WIDE OPEN CRACK DETECTED AS CLOSED! BECAUSE
C THE STIFFNESS IS ARRESTED AT ECONC/1000. AND THE STRESS IS
C NEGETIVE. THEREFORE NEGETIVE CRACK STRAINS!
C
            IF (CRACK(L,87+ICRACK,J) .LE. EINITAL/1000.0) GO TO 140
            IF (ICRACK .EQ. NCRACK) THEN
                CRACK(L,1,J) = CRACK(L,1,J) - 1.000
                GO TO 160
            ENDIF
C
            CALL ARANGE (ICRACK, NCRACK, L, J)
        ENDIF
  140   CONTINUE
  150 CONTINUE
C
C             INITIALIZE THE CONSTITUTIVE MATRIX FOR THE INTACT CONC.
C
  160 CONTINUE
      IFG = 2
      CALL PSTSTN (SIGMA, PVAL, PDIR, IFG, IOUT)
      DO 180 II = 1, 6
          DO 170 JJ = 1, 6
              DEP(II,JJ) = 0.0
  170     CONTINUE
  180 CONTINUE
C
C             RETRIEVE THE CONSTITUTIVE MATRIX (IN PRINCIPAL AXES)
C
      ES = HISTO(L,20,J)
      NUS = HISTO(L,21,J)
      CONST = ES/((1.-2.0*NUS)*(1+NUS))
C     WRITE(17,818)ES,NUS
C818   FORMAT(1X,'ES=',D12.5,'NUS',D12.5)
      DEP(1,1) = CONST*(1.0-NUS)
      DEP(2,2) = CONST*(1.0-NUS)
      DEP(3,3) = CONST*(1.0-NUS)
      DEP(1,2) = CONST*NUS
      DEP(2,1) = DEP(1,2)
      DEP(3,1) = DEP(1,2)
      DEP(1,3) = DEP(1,2)
      DEP(2,3) = DEP(1,2)
      DEP(3,2) = DEP(1,2)
      DEP(4,4) = ES/(2.0*(1+NUS))
      DEP(5,5) = DEP(4,4)
      DEP(6,6) = DEP(4,4)
C             INVERT THE 'DEP' MATRIX.
C
C             N = NO. OF ROWS IN 'DEP'
C           NRC = MAX. NO. OF ROWS(AND COLUMNS) ALLOWED.
C           EPS = MIN. ALLOWABLE MAGNITUDE FOR A PIVOT ELEMENT.
C
C
```

```
      N = 6
      NRC = 6
      EPS = 10.0E-20
C
C              INVERSE OF 'DEP' WILL BE RETURNED IN 'DEP'.
C
C              DETER = DETERMINANT OF 'DEP' THAT IS RETURNED AS THE
C                      VALUE OF FUNCTION 'SIMUL'.
C
C
      DETER = SIMUL(N,DEP,EPS,NRC,IOUT)
      IF (DETER .EQ. 0.0) THEN
          WRITE (IOUT, 900) L, J, ELNUM
          STOP
      ENDIF
C
C              DETERMINE PRINCIPAL STRAINS CONTRIBUTED TO THE TOTAL
C              STRAINS BY THE INTACT CONCRETE.
C
      SIGMA1 = PVAL(1)
      SIGMA2 = PVAL(2)
      SIGMA3 = PVAL(3)
      SMAX = DMAX1(DMAX1(DABS(SIGMA1),DABS(SIGMA2)),DABS(SIGMA3))
      IF (DABS(SIGMA1) .LE. SMAX*1.E-4) SIGMA1 = 0.0
      IF (DABS(SIGMA2) .LE. SMAX*1.E-4) SIGMA2 = 0.0
      IF (DABS(SIGMA3) .LE. SMAX*1.E-4) SIGMA3 = 0.0
      SIGMA1 = DMIN1(PROPER(6),SIGMA1)
      EPS1 = DEP(1,1)*SIGMA1 + DEP(1,2)*SIGMA2 + DEP(1,3)*SIGMA3
      EPS2 = DEP(2,1)*SIGMA1 + DEP(2,2)*SIGMA2 + DEP(2,3)*SIGMA3
      EPS3 = DEP(3,1)*SIGMA1 + DEP(3,2)*SIGMA2 + DEP(3,3)*SIGMA3
      RETURN
  900    FORMAT(//,1X,'    ERROR',/,5X,'NEGATIVE PIVOT ENCOUNTERED'
     1          ,' DURING INVERSION OF CONSTITUTIVE MATRIX FOR INTACT '
     2          ,'CONCRETE ...CRKPNT  STOP 3','G.PT',I2,'LAYER',I2,
     3          'ELNUM',I2)
      END
C
C
C
C *****************************************************************
C     INCLUDE(PROCESS)
      SUBROUTINE TSSTIF(EPSX,EPSY,EPSZ,GAMAXY,GAMAYZ,GAMAXZ,SIGMA1,
     $SIGMA2,SIGMA3,DTSTIF,L,J,CONVER,NSTIFF,CHKGUS,PROPER,TOLER,
     $ELNUM,NNEL,IFG,IFC,ICOUP,IFLAG1,IFLAG2,IFLAG3,ISOFT,ITCRK,TEMP11)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C DTRANS    SPGN
C
C*************************************************************
      INTEGER ELNUM
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),INISTY,IHIST1
     $,IHIST2
      COMMON/ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      DIMENSION PROPER(25),DTSTIF(6,6),TENXYZ(6,6),TEMP11(6,6)
      REAL*8 NCR(3,3),BSTL(3,3),NT(3,3),N(3,3)
      LOGICAL CHKGUS
C
      DO 110 IP = 1, 6
```

```
        DO 100 KP = 1, 6
            DTSTIF(IP,KP) = 0.0
 100    CONTINUE
 110 CONTINUE
     DO 130 IM = 1, 6
        DO 120 IN = 1, 6
            TEMP11(IM,IN) = 0.0
            TENXYZ(IM,IN) = 0.0
 120    CONTINUE
 130 CONTINUE
C
     ICOUP = 1
     IFLAG1 = 0
     IFLAG2 = 0
     IFLAG3 = 0
     IFLAG4 = 0
     IFLAG5 = 0
     IFLAG6 = 0
     ICHECK = 0
     MCHECK = 0
     F3 = 0.0
     F2 = 0.0
     F1 = 0.0
     ISPG2 = 0
     ISPG3 = 0
C
     DO 200 ISP = 1, 3
C
C
        ISP2 = 0
        ISP3 = 0
C        WRITE(*,*) 'FLAGS 1',IFLAG1,'FLAG2',IFLAG2,'FLAG3',IFLAG3
        IF (TENSTF(L,ISP,J) .EQ. 1.0) THEN
            MCHECK = 1
C             WRITE(*,*) 'MCHECK=1 ISP',ISP,'ISPG',TENSTF(L,3+ISP,J)
            IF (ISP .EQ. 1) IFLAG1 = 1
            IF (ISP .EQ. 2) IFLAG2 = 1
            IF (ISP .EQ. 3) IFLAG3 = 1
            GO TO 190
        ENDIF
C
C CHECK THE STATUS OF PRESENT BUT INACTIVE TENSIONSTIFFENING SPRINGS
C
C ... THE CURRENT PRINCIPAL STRESS DIRECTIONS ARE--
C
        N(1,1) = HISTO(L,10,J)
        N(2,1) = HISTO(L,11,J)
        N(3,1) = HISTO(L,12,J)
        N(1,2) = HISTO(L,13,J)
        N(2,2) = HISTO(L,14,J)
        N(3,2) = HISTO(L,15,J)
        N(1,3) = HISTO(L,16,J)
        N(2,3) = HISTO(L,17,J)
        N(3,3) = HISTO(L,18,J)
C
        IF (TENSTF(L,ISP,J) .EQ. 0.0) THEN
            ISPG = TENSTF(L,3+ISP,J)
C
            IF (ISPG .EQ. 0) THEN
                IF (ISP .EQ. 1) IFLAG1 = 1
                IF (ISP .EQ. 2) IFLAG2 = 1
                IF (ISP .EQ. 3) IFLAG3 = 1
                GO TO 190
```

```
              ENDIF
C
C RECOVER THE SPRING DIRECTION AND THE STRAINS IN THAT DIRECTION
C
              ISPDCS = 9*(ISPG-1) + 1
              AL1 = TENSTF(L,30+ISPDCS,J)
              AM1 = TENSTF(L,31+ISPDCS,J)
              AN1 = TENSTF(L,32+ISPDCS,J)
C STEEL DIRECTION STRAINS ARE....
              EPSSTL = EPSX*AL1*AL1 + EPSY*AM1*AM1 + EPSZ*AN1*AN1 +
     1            GAMAXY*AL1*AM1 + GAMAYZ*AM1*AN1 + GAMAXZ*AN1*AL1
              IF (EPSSTL .GT. 0.0) THEN
                  TENSTF(L,69+ISPG,J) = EPSSTL
              ELSE
                  TENSTF(L,69+ISPG,J) = 0.0
              ENDIF
C CRACK STRAIN IS
              EPSCR = TENSTF(L,9+ISPG,J)
C            WRITE(*,*) 'TSSTIF EPSSTL1',EPSSTL,'EPSCR',EPSCR,'SPG',ISPG
C
C
C CHECK IF ANY ACTIVE CRACK CONTRIBUTES TO ACTIVATING THE T.SPRING
C
              NCRACK = INT(CRACK(L,1,J))
              DO 180 ICK = 1, NCRACK
                  IMG = 9*(ICK-1) + 1
                  NCR(1,1) = CRACK(L,137+IMG,J)
                  NCR(1,2) = CRACK(L,138+IMG,J)
                  NCR(1,3) = CRACK(L,139+IMG,J)
                  NCR(2,1) = CRACK(L,140+IMG,J)
                  NCR(2,2) = CRACK(L,141+IMG,J)
                  NCR(2,3) = CRACK(L,142+IMG,J)
                  NCR(3,1) = CRACK(L,143+IMG,J)
                  NCR(3,2) = CRACK(L,144+IMG,J)
                  NCR(3,3) = CRACK(L,145+IMG,J)
                  EPSCRK = CRACK(L,57+ICK,J)
                  EPSULT = CRACK(L,127+ICK,J)
                  EPSCRK = DMIN1(EPSULT,EPSCRK)
                  SIG1N = CRACK(L,37+ICK,J)*(1.0-0.2*EPSCRK/EPSULT)
                  N(1,1) = TENSTF(L,ISPDCS+30,J)
                  N(1,2) = TENSTF(L,ISPDCS+31,J)
                  N(1,3) = TENSTF(L,ISPDCS+32,J)
                  N(2,1) = TENSTF(L,ISPDCS+33,J)
                  N(2,2) = TENSTF(L,ISPDCS+34,J)
                  N(2,3) = TENSTF(L,ISPDCS+35,J)
                  N(3,1) = TENSTF(L,ISPDCS+36,J)
                  N(3,2) = TENSTF(L,ISPDCS+37,J)
                  N(3,3) = TENSTF(L,ISPDCS+38,J)
                  DO 150 IMM = 1, 3
                      DO 140 IMP = 1, 3
                          BSTL(IMM,IMP) = 0.0
                          BSTL(IMM,IMP) = BSTL(IMM,IMP) + N(IMM,1)*NCR(
     1                        IMP,1)
                          BSTL(IMM,IMP) = BSTL(IMM,IMP) + N(IMM,2)*NCR(
     1                        IMP,2)
                          BSTL(IMM,IMP) = BSTL(IMM,IMP) + N(IMM,3)*NCR(
     1                        IMP,3)
  140                 CONTINUE
  150             CONTINUE
                  DO KJ = 1, 3
                      DO KR = 1, 3
                          IF (DABS(BSTL(KJ,KR)) .LE. 0.01) BSTL(KJ,KR)
     1                         = 0.0
```

```
               END DO
            END DO
            AL1 = BSTL(1,1)
            AM1 = BSTL(2,1)
            AN1 = BSTL(3,1)
            AL2 = BSTL(1,2)
            AM2 = BSTL(2,2)
            AN2 = BSTL(3,2)
            AL3 = BSTL(1,3)
            AM3 = BSTL(2,3)
            AN3 = BSTL(3,3)
            DOT=NCR(1,1)*N(1,1)+NCR(1,2)*N(1,2)+NCR(1,3)*N(1,3)
            IF (DABS(DOT) .GE. 0.250) THEN
C
C CHECK FOR SPECIAL CASES.....
C
               IF (TENSTF(L,4,J).EQ.0 .OR. TENSTF(L,5,J).EQ.0
     1              .OR. TENSTF(L,6,J).EQ.0) THEN
                  IF (DABS(AN3) .LE. 0.01) THEN
                     AL1 = BSTL(1,1)
                     AM1 = BSTL(2,1)
                     AN1 = BSTL(3,1)
                     AL2 = BSTL(1,3)
                     AM2 = BSTL(2,3)
                     AN2 = BSTL(3,3)
                     AL3 = BSTL(1,2)
                     AM3 = BSTL(2,2)
                     AN3 = BSTL(3,2)
                  ENDIF
               ENDIF
C
               IF (TENSTF(L,4,J).EQ.0 .AND. TENSTF(L,5,J).EQ.0
     1              .OR. TENSTF(L,5,J).EQ.0 .AND. TENSTF(L,6,J)
     2             .EQ.0 .OR. TENSTF(L,6,J).EQ.0 .AND. TENSTF(L,4
     3             ,J).EQ.0) THEN
                  IF (DABS(AM2) .LE. 0.01) THEN
                     AL1 = BSTL(1,1)
                     AL2 = BSTL(1,3)
                     AL3 = BSTL(1,2)
                     AM1 = BSTL(2,1)
                     AM2 = BSTL(2,3)
                     AM3 = BSTL(2,2)
                     AN1 = BSTL(3,1)
                     AN2 = BSTL(3,3)
                     AN3 = BSTL(3,2)
                  ENDIF
               ENDIF
C
               SIG1NN = 0.0
               IF (TENSTF(L,4,J).GT.0.0 .AND. TENSTF(L,5,J).GT.
     1             0.0 .AND. TENSTF(L,6,J).GT.0.0) THEN
                  SIG1NN = SIG1N
               ELSE IF (TENSTF(L,4,J).GT.0.0 .AND. TENSTF(L,5,J)
     1                 .GT.0.0 .AND. TENSTF(L,6,J).EQ.0.0 .OR.
     2                 TENSTF(L,5,J).GT.0.0 .AND. TENSTF(L,6,J)
     3                 .GT.0.0 .AND. TENSTF(L,4,J).EQ.0.0 .OR.
     4                 TENSTF(L,4,J).GT.0.0 .AND. TENSTF(L,6,J)
     5                 .GT.0.0 .AND. TENSTF(L,5,J).EQ.0.0) THEN
                  SIG1NN = SIG1N*((AL3*AN1-AL1*AN3)*(AM2*AN3-AM3
     1                 *AN2)-(AN1*AM3-AM1*AN3)*(AL2*AN3-AN2*AL3))
     2                 /(AN3*(AM1*(AL2*AN3-AN2*AL3)-AL1*(AM2*AN3-
     3                 AM3*AN2)))
               ELSE IF (TENSTF(L,4,J).GT.0.0 .AND. TENSTF(L,5,J)
```

```
     1                   .EQ.0.0 .AND. TENSTF(L,6,J).EQ.0.0 .OR.
     2                   TENSTF(L,5,J).GT.0.0 .AND. TENSTF(L,6,J)
     3                   .EQ.0.0 .AND. TENSTF(L,4,J).EQ.0.0 .OR.
     4                   TENSTF(L,4,J).GT.0.0 .AND. TENSTF(L,6,J)
     5                   .EQ.0.0 .AND. TENSTF(L,5,J).EQ.0.0) THEN
                      SIG1NN = SIG1N*((AN1*AM2-AM1*AN2)*(AL3*AM2-AM3
     1                   *AL2)-(AL2*AM1-AM2*AL1)*(AM3*AN2-AN3*AM2))
     2                   /(AL1*AM2*(AM3*AN2-AN3*AM2))
                  ENDIF
                  IF (TENSTF(L,ISP,J) .EQ. 0.0) THEN
                      IF (EPSSTL .GE. EPSCR) THEN
                          IF (ISP .EQ. 1) ICK1 = ICK
                          IF (ISP .EQ. 2) ICK2 = ICK
                          IF (ISP .EQ. 3) ICK3 = ICK
                          ICHECK = 1
                          ICOUP = 0
                          MCHECK = 0
                          IF (ISP .EQ. 1) IFLAG1 = 1
                          IF (ISP .EQ. 2) IFLAG2 = 1
                          IF (ISP .EQ. 3) IFLAG3 = 1
                          TENSTF(L,ISP,J) = 1.0
                          TENSTF(L,6+ISPG,J) = SIG1NN


                          TENSTF(L,15+ISPG,J) = EPSCR
                          TENSTF(L,18+ISPG,J) = EPSCR
                          TENSTF(L,24+ISPG,J) = TENSTF(L,6+ISPG,J)/
     1                        EPSCR
                      ELSE IF (ISOFT .EQ. 1) THEN
                          MCHECK = 0
                          ICHECK = 1
                          ICOUP = 0
                          IF (ISP .EQ. 1) IFLAG4 = 2
                          IF (ISP .EQ. 2) IFLAG5 = 2
                          IF (ISP .EQ. 3) IFLAG6 = 2
                          TENSTF(L,6+ISPG,J) = SIG1NN
                          TENSTF(L,15+ISPG,J) = EPSSTL
                          TENSTF(L,18+ISPG,J) = EPSSTL
                          IF (EPSSTL .GT. 0.0) TENSTF(L,24+ISPG,J)
     1                         = SIG1NN/EPSSTL
                      ELSE IF (EPSSTL.GT.0.0 .AND. EPSSTL.LT.EPSCR)
     1                        THEN
                          MCHECK = 0
                          ICHECK = 1
                          ICOUP = 0
                          IF (ISP .EQ. 1) IFLAG4 = 2
                          IF (ISP .EQ. 2) IFLAG5 = 2
                          IF (ISP .EQ. 3) IFLAG6 = 2
                          TENSTF(L,6+ISPG,J) = SIG1NN
                          TENSTF(L,15+ISPG,J) = EPSSTL
                          TENSTF(L,18+ISPG,J) = EPSSTL
                          IF (EPSSTL .GT. 0.0) TENSTF(L,24+ISPG,J)
     1                         = SIG1NN/EPSSTL
                      ENDIF
                  ENDIF
              ENDIF
  180         CONTINUE
          ENDIF
  190     CONTINUE
  200 CONTINUE
C
C ... IF THE CURRENT CRACK IS NOT STRAIN SOFTENING THEN
C
      NCRACK = INT(CRACK(L,1,J))
```

```
      IF (MCHECK .EQ. 0) THEN
          NCRACK = INT(CRACK(L,1,J))
          IF (IFLAG1.EQ.1 .AND. IFLAG2.EQ.1 .AND. IFLAG3.EQ.1) THEN
              ITCRK = NCRACK
          ELSE IF (IFLAG1.EQ.1 .AND. IFLAG2.EQ.1 .AND. IFLAG3.EQ.0 .OR.
     1            IFLAG1.EQ.1 .AND. IFLAG3.EQ.1 .AND. IFLAG2.EQ.0 .OR.
     2            IFLAG2.EQ.1 .AND. IFLAG3.EQ.1 .AND. IFLAG1.EQ.0) THEN
              IF (IFLAG4.EQ.2 .OR. IFLAG5.EQ.2 .OR. IFLAG6.EQ.2) THEN
                  IF (TENSTF(L,1,J).EQ.1 .AND. IFLAG1.EQ.1 .OR. TENSTF(L
     1                ,2,J).EQ.1 .AND. IFLAG2.EQ.1 .OR. TENSTF(L,3,J)
     2                .EQ.1 .AND. IFLAG3.EQ.1) THEN
                      IF (IFLAG4 .EQ. 2) TENSTF(L,1,J) = 1.0
                      IF (IFLAG5 .EQ. 2) TENSTF(L,2,J) = 1.0
                      IF (IFLAG6 .EQ. 2) TENSTF(L,3,J) = 1.0
                      ITCRK = NCRACK
                      IFLAG1 = 1
                      IFLAG2 = 1
                      IFLAG3 = 1
                  ENDIF
C                WRITE(*,*)'ALL FLAGS ON CASE A'
              ELSE IF (TENSTF(L,4,J).EQ.0.0 .OR. TENSTF(L,5,J).EQ.0.0
     1                 .OR. TENSTF(L,6,J).EQ.0.0) THEN
                  ITCRK = NCRACK
                  IFLAG1 = 1
                  IFLAG2 = 1
                  IFLAG3 = 1
C                WRITE(*,*)'ALL FLAGS ON CASE B'
              ENDIF
          ELSE IF (IFLAG1.EQ.1 .AND. IFLAG2.EQ.0 .AND. IFLAG3.EQ.0 .OR.
     1            IFLAG2.EQ.1 .AND. IFLAG3.EQ.0 .AND. IFLAG1.EQ.0 .OR.
     2            IFLAG3.EQ.1 .AND. IFLAG1.EQ.0 .AND. IFLAG2.EQ.0) THEN
              IF (IFLAG4.EQ.2 .AND. IFLAG5.EQ.2 .AND. IFLAG6.EQ.0 .OR.
     1            IFLAG5.EQ.2 .AND. IFLAG6.EQ.2 .AND. IFLAG4.EQ.0 .OR.
     2            IFLAG6.EQ.2 .AND. IFLAG4.EQ.2 .AND. IFLAG5.EQ.0) THEN
                  IF (IFLAG3.EQ.1 .AND. TENSTF(L,6,J).GT.0.0 .OR. IFLAG2
     1                .EQ.1 .AND. TENSTF(L,5,J).GT.0.0 .OR. IFLAG1.EQ.1
     2                 .AND. TENSTF(L,4,J).GT.0.0) THEN
                      ITCRK = NCRACK
                      IF (IFLAG4 .EQ. 2) TENSTF(L,1,J) = 1.0
                      IF (IFLAG5 .EQ. 2) TENSTF(L,2,J) = 1.0
                      IF (IFLAG6 .EQ. 2) TENSTF(L,3,J) = 1.0
                      IFLAG1 = 1
                      IFLAG2 = 1
                      IFLAG3 = 1
                  ENDIF
              ELSE IF (TENSTF(L,4,J).EQ.1.0 .AND. TENSTF(L,5,J).EQ.0.0
     1                 .AND. TENSTF(L,6,J).EQ.0.0 .OR. TENSTF(L,5,J).EQ.
     2                1.0 .AND. TENSTF(L,4,J).EQ.0.0 .AND. TENSTF(L,6,J)
     3                .EQ.0.0 .OR. TENSTF(L,6,J).EQ.1.0 .AND. TENSTF(L,5
     4                ,J).EQ.0.0 .AND. TENSTF(L,4,J).EQ.0.0) THEN
                  ITCRK = NCRACK
                  IFLAG1 = 1
                  IFLAG2 = 1
                  IFLAG3 = 1
              ENDIF
          ENDIF
      ENDIF
      IF (IFLAG1.EQ.1 .AND. IFLAG2.EQ.1 .AND. IFLAG3.EQ.1) THEN
          IF (ICHECK .EQ. 1) THEN
              CONVER = 999.0
              WRITE (*, *) 'TSTIF 1 999 ELNUM', ELNUM
              NSTIFF = 1
              CHKGUS = .TRUE.
```

```
      IMG = 0
      DO 290 ISP = 1, 3
          ISPG = TENSTF(L,3+ISP,J)
          DO 220 IM = 1, 6
              DO 210 IN = 1, 6
                  TENXYZ(IM,IN) = 0.0
  210         CONTINUE
  220     CONTINUE
          IF (ISPG.GT.0 .AND. TENSTF(L,ISP,J).EQ.1.0) THEN
              ISMG = (ISPG-1)*9 + 1
C
C ... TENSION STIFFENING (STEEL) DIRECTION W.R.T TO 'GLOBAL'
C
              N(1,1) = TENSTF(L,ISMG+30,J)
              N(1,2) = TENSTF(L,ISMG+31,J)
              N(1,3) = TENSTF(L,ISMG+32,J)
              N(2,1) = TENSTF(L,ISMG+33,J)
              N(2,2) = TENSTF(L,ISMG+34,J)
              N(2,3) = TENSTF(L,ISMG+35,J)
              N(3,1) = TENSTF(L,ISMG+36,J)
              N(3,2) = TENSTF(L,ISMG+37,J)
              N(3,3) = TENSTF(L,ISMG+38,J)
              EPSTL1 = EPSX*N(1,1)*N(1,1) + EPSY*N(1,2)*N(1,2)
     1             + EPSZ*N(1,3)*N(1,3) + GAMAXY*N(1,1)*N(1,2)
     2             + GAMAYZ*N(1,2)*N(1,3) + GAMAXZ*N(1,3)*N(1,1)
              EPSTL2 = EPSX*N(2,1)*N(2,1) + EPSY*N(2,2)*N(2,2)
     1             + EPSZ*N(2,3)*N(2,3) + GAMAXY*N(2,1)*N(2,2)
     2             + GAMAYZ*N(2,2)*N(2,3) + GAMAXZ*N(2,3)*N(2,1)
              EPSTL3 = EPSX*N(3,1)*N(3,1) + EPSY*N(3,2)*N(3,2)
     1             + EPSZ*N(3,3)*N(3,3) + GAMAXY*N(3,1)*N(3,2)
     2             + GAMAYZ*N(3,2)*N(3,3) + GAMAXZ*N(3,3)*N(3,1)
C
              NCR(1,1) = HISTO(L,44,J)
              NCR(1,2) = HISTO(L,45,J)
              NCR(1,3) = HISTO(L,46,J)
              NCR(2,1) = HISTO(L,47,J)
              NCR(2,2) = HISTO(L,48,J)
              NCR(2,3) = HISTO(L,49,J)
              NCR(3,1) = HISTO(L,50,J)
              NCR(3,2) = HISTO(L,51,J)
              NCR(3,3) = HISTO(L,52,J)
              DO 240 IMM = 1, 3
                  DO 230 IMP = 1, 3
                      BSTL(IMM,IMP) = 0.0
                      BSTL(IMM,IMP) = BSTL(IMM,IMP) + N(IMM,1)*
     1                    NCR(IMP,1)
                      BSTL(IMM,IMP) = BSTL(IMM,IMP) + N(IMM,2)*
     1                    NCR(IMP,2)
                      BSTL(IMM,IMP) = BSTL(IMM,IMP) + N(IMM,3)*
     1                    NCR(IMP,3)
  230             CONTINUE
  240         CONTINUE
              DO MN = 1, 3
                  DO MJ = 1, 3
                      IF (DABS(BSTL(MN,MJ)) .LE. 0.01) BSTL(MN,
     1                    MJ) = 0.0
                  END DO
              END DO
              AL1 = BSTL(1,1)
              AM1 = BSTL(1,2)
              AN1 = BSTL(1,3)
              AL2 = BSTL(2,1)
              AM2 = BSTL(2,2)
```

```
                AN2 = BSTL(2,3)
                AL3 = BSTL(3,1)
                AM3 = BSTL(3,2)
                AN3 = BSTL(3,3)
                PEPMAX = HISTO(L,41,J)
                PEPMID = HISTO(L,42,J)
                PEPMIN = HISTO(L,43,J)
                PSTL1 = PEPMAX*AL1*AL1 + PEPMID*AM1*AM1 + PEPMIN*
     1              AN1*AN1
                PSTL2 = PEPMAX*AL2*AL2 + PEPMID*AM2*AM2 + PEPMIN*
     1              AN2*AN2
                PSTL3 = PEPMAX*AL3*AL3 + PEPMID*AM3*AM3 + PEPMIN*
     1              AN3*AN3
                TENXYZ(1,1) = TENSTF(L,24+ISPG,J)
                IMG = IMG + 1
                IF (IMG .EQ. 1) THEN
                    IF (ICOUP .EQ. 1) THEN
                        ISPG2 = 0
                        ISPG3 = 0
                        IF (ISPG .EQ. 1) THEN
                            ISPG1 = ISPG
                            ISPG2 = 2
                            ISPG3 = 3
                        ELSE IF (ISPG .EQ. 2) THEN
                            ISPG1 = ISPG
                            ISPG2 = 3
                            ISPG3 = 1
                        ELSE IF (ISPG .EQ. 3) THEN
                            ISPG1 = ISPG
                            ISPG2 = 1
                            ISPG3 = 2
                        ENDIF
                        ISP2 = 0
                        ISP3 = 0
                        IF (TENSTF(L,4,J) .EQ. ISPG2) ISP2 = 1
                        IF (TENSTF(L,5,J) .EQ. ISPG2) ISP2 = 2
                        IF (TENSTF(L,6,J) .EQ. ISPG2) ISP2 = 3
                        IF (TENSTF(L,4,J) .EQ. ISPG3) ISP3 = 1
                        IF (TENSTF(L,5,J) .EQ. ISPG3) ISP3 = 2
                        IF (TENSTF(L,6,J) .EQ. ISPG3) ISP3 = 3
                        CNST12 = 0.0
                        CNST23 = 0.0
                        CNST13 = 0.0
C
                        IF (TENSTF(L,ISP2,J) .EQ. 0) THEN
                            IF (PSTL2.LE.TENSTF(L,9+ISPG,J) .AND.
     1                          EPSTL2.GT.TENSTF(L,9+ISPG,J)) THEN
                            CHKGUS = .TRUE.
                            CONVER = 999.0
                            WRITE (*, *) 'TSTIF 2 999 ELNUM',
     1                          ELNUM
                            ENDIF
                        ENDIF
C
                        IF (TENSTF(L,ISP3,J) .EQ. 0) THEN
                            IF (PSTL3.LE.TENSTF(L,9+ISPG,J) .AND.
     1                          EPSTL3.GT.TENSTF(L,9+ISPG,J)) THEN
                            CHKGUS = .TRUE.
                            CONVER = 999.0
                            WRITE (*, *) 'TSTIF 3 999 ELNUM',
     1                          ELNUM
                            ENDIF
                        ENDIF
```

```
c
                    IF (TENSTF(L,ISP,J).EQ.1.0 .AND. TENSTF(L,
     1                  ISP2,J).EQ.1.0) THEN
                        CNST12 = DSQRT(TENSTF(L,24+ISPG,J)/
     1                      HISTO(L,26,J)*TENSTF(L,24+ISPG2,J)
     2                      /HISTO(L,26,J))
                    ELSE IF (TENSTF(L,ISP,J).EQ.1.0 .AND.
     1                      TENSTF(L,ISP2,J).EQ.0.0) THEN
                        CNST12 = DSQRT(TENSTF(L,24+ISPG,J)/
     1                      HISTO(L,26,J)*HISTO(L,20,J)/HISTO(
     2                      L,26,J))
                        IF (EPSTL2 .GT. TENSTF(L,9+ISPG,J))
     1                      CNST12 = 0.0
                    ELSE IF (TENSTF(L,ISP,J).EQ.0.0 .AND.
     1                      TENSTF(L,ISP2,J).EQ.1.0) THEN
                        CNST12 = DSQRT(TENSTF(L,24+ISPG2,J)/
     1                      HISTO(L,26,J)*HISTO(L,20,J)/HISTO(
     2                      L,26,J))
                        IF (EPSTL1 .GT. TENSTF(L,9+ISPG,J))
     1                      CNST12 = 0.0
                    ELSE IF (TENSTF(L,ISP,J).EQ.0.0 .AND.
     1                      TENSTF(L,ISP2,J).EQ.0.0) THEN
                        CNST12 = DSQRT(HISTO(L,20,J)/HISTO(L,
     1                      26,J)*HISTO(L,20,J)/HISTO(L,26,J))
                        IF (EPSTL1.GT.TENSTF(L,9+ISPG,J) .OR.
     1                      EPSTL2.GT.TENSTF(L,9+ISPG,J))
     2                      CNST12 = 0.0
                    ENDIF
                    IF (TENSTF(L,ISP,J).EQ.1.0 .AND. TENSTF(L,
     1                  ISP3,J).EQ.1.0) THEN
                        CNST13 = DSQRT(TENSTF(L,24+ISPG,J)/
     1                      HISTO(L,26,J)*TENSTF(L,24+ISPG3,J)
     2                      /HISTO(L,26,J))
                    ELSE IF (TENSTF(L,ISP,J).EQ.1.0 .AND.
     1                      TENSTF(L,ISP3,J).EQ.0.0) THEN
                        CNST13 = DSQRT(TENSTF(L,24+ISPG,J)/
     1                      HISTO(L,26,J)*HISTO(L,20,J)/HISTO(
     2                      L,26,J))
                        IF (EPSTL3 .GT. TENSTF(L,9+ISPG,J))
     1                      CNST13 = 0.0
                    ELSE IF (TENSTF(L,ISP,J).EQ.0.0 .AND.
     1                      TENSTF(L,ISP3,J).EQ.1.0) THEN
                        CNST13 = DSQRT(TENSTF(L,24+ISPG3,J)/
     1                      HISTO(L,26,J)*HISTO(L,20,J)/HISTO(
     2                      L,26,J))
                        IF (EPSTL1 .GT. TENSTF(L,9+ISPG,J))
     1                      CNST13 = 0.0
                    ELSE IF (TENSTF(L,ISP,J).EQ.0.0 .AND.
     1                      TENSTF(L,ISP3,J).EQ.0.0) THEN
                        CNST13 = DSQRT(HISTO(L,20,J)/HISTO(L,
     1                      26,J)*HISTO(L,20,J)/HISTO(L,26,J))
                        IF (EPSTL1.GT.TENSTF(L,9+ISPG,J) .OR.
     1                      EPSTL3.GT.TENSTF(L,9+ISPG,J))
     2                      CNST13 = 0.0
                    ENDIF
                    IF (TENSTF(L,ISP3,J).EQ.1.0 .AND. TENSTF(L
     1                  ,ISP2,J).EQ.1.0) THEN
                        CNST23 = DSQRT(TENSTF(L,24+ISPG3,J)/
     1                      HISTO(L,26,J)*TENSTF(L,24+ISPG2,J)
     2                      /HISTO(L,26,J))
                    ELSE IF (TENSTF(L,ISP3,J).EQ.1.0 .AND.
     1                      TENSTF(L,ISP2,J).EQ.0.0) THEN
                        CNST23 = DSQRT(TENSTF(L,24+ISPG3,J)/
```

```
     1                          HISTO(L,26,J)*HISTO(L,20,J)/HISTO(
     2                          L,26,J))
                            IF (EPSTL2 .GT. TENSTF(L,9+ISPG,J))
     1                          CNST23 = 0.0
                        ELSE IF (TENSTF(L,ISP3,J).EQ.0.0 .AND.
     1                          TENSTF(L,ISP2,J).EQ.1.0) THEN
                            CNST23 = DSQRT(TENSTF(L,24+ISPG2,J)/
     1                          HISTO(L,26,J)*HISTO(L,20,J)/HISTO(
     2                          L,26,J))
                            IF (EPSTL3 .GT. TENSTF(L,9+ISPG,J))
     1                          CNST23 = 0.0
                        ELSE IF (TENSTF(L,ISP2,J).EQ.0.0 .AND.
     1                          TENSTF(L,ISP3,J).EQ.0.0) THEN
                            CNST23 = DSQRT(HISTO(L,20,J)/HISTO(L,
     1                          26,J)*HISTO(L,20,J)/HISTO(L,26,J))
                            IF (EPSTL2.GT.TENSTF(L,9+ISPG,J) .OR.
     1                          EPSTL3.GT.TENSTF(L,9+ISPG,J))
     2                          CNST23 = 0.0
                        ENDIF
                        TEMP11(2,1) = CNST12*HISTO(L,26,J)*HISTO(L
     1                      ,27,J)
                        TEMP11(1,2) = CNST12*HISTO(L,26,J)*HISTO(L
     1                      ,27,J)
                        TEMP11(3,1) = CNST13*HISTO(L,26,J)*HISTO(L
     1                      ,27,J)
                        TEMP11(1,3) = CNST13*HISTO(L,26,J)*HISTO(L
     1                      ,27,J)
                        TEMP11(2,3) = CNST23*HISTO(L,26,J)*HISTO(L
     1                      ,27,J)
                        TEMP11(3,2) = CNST23*HISTO(L,26,J)*HISTO(L
     1                      ,27,J)
                        CALL DTRANS (TEMP11, N, IOUT)
                        ICOUP = 1
                    ENDIF
                ENDIF
C                  END IF
                CALL DTRANS (TENXYZ, N, IOUT)
            ENDIF
            DO 280 IJ = 1, 6
                DO 270 JK = 1, 6
                    DTSTIF(IJ,JK) = TENXYZ(IJ,JK) + DTSTIF(IJ,JK)
  270           CONTINUE
  280       CONTINUE
  290   CONTINUE
        ENDIF
    ENDIF
    IF (ICHECK .EQ. 1) RETURN
    IMG = 0
    ICOUP = 1
    DO 420 ISP = 1, 3
        DO 310 IP = 1, 6
            DO 300 KP = 1, 6
                TENXYZ(IP,KP) = 0.0
  300       CONTINUE
  310   CONTINUE
C
C ... THE CURRENT PRINCIPAL STRESS DIRECTIONS ARE--
C
        N(1,1) = HISTO(L,10,J)
        N(2,1) = HISTO(L,11,J)
        N(3,1) = HISTO(L,12,J)
        N(1,2) = HISTO(L,13,J)
        N(2,2) = HISTO(L,14,J)
```

```
      N(3,2) = HISTO(L,15,J)
      N(1,3) = HISTO(L,16,J)
      N(2,3) = HISTO(L,17,J)
      N(3,3) = HISTO(L,18,J)
      IF (TENSTF(L,ISP,J) .EQ. 1.0) THEN
          IFLAG1 = 1
          IFLAG2 = 1
          IFLAG3 = 1
          ISPG = TENSTF(L,3+ISP,J)
          ISPDCS = 9*(ISPG-1) + 1
          AL1 = TENSTF(L,30+ISPDCS,J)
          AM1 = TENSTF(L,31+ISPDCS,J)
          AN1 = TENSTF(L,32+ISPDCS,J)
          EPSSTL = EPSX*AL1*AL1 + EPSY*AM1*AM1 + EPSZ*AN1*AN1 +
     1        GAMAXY*AL1*AM1 + GAMAYZ*AM1*AN1 + GAMAXZ*AN1*AL1
          IMMG = (ISPG-1)*9 + 1
          NT(1,1) = TENSTF(L,30+IMMG,J)
          NT(1,2) = TENSTF(L,31+IMMG,J)
          NT(1,3) = TENSTF(L,32+IMMG,J)
          NT(2,1) = TENSTF(L,33+IMMG,J)
          NT(2,2) = TENSTF(L,34+IMMG,J)
          NT(2,3) = TENSTF(L,35+IMMG,J)
          NT(3,1) = TENSTF(L,36+IMMG,J)
          NT(3,2) = TENSTF(L,37+IMMG,J)
          NT(3,3) = TENSTF(L,38+IMMG,J)
          DO 330 IM = 1, 3
              DO 320 IP = 1, 3
                  BSTL(IM,IP) = 0.0
                  BSTL(IM,IP) = BSTL(IM,IP) + NT(IM,1)*N(1,IP)
                  BSTL(IM,IP) = BSTL(IM,IP) + NT(IM,2)*N(2,IP)
                  BSTL(IM,IP) = BSTL(IM,IP) + NT(IM,3)*N(3,IP)
320           CONTINUE
330       CONTINUE
          DO IR = 1, 3
              DO MR = 1, 3
                  IF (DABS(BSTL(IR,MR)) .LE. 0.01) BSTL(IR,MR) = 0.0
              END DO
          END DO
          AL2 = BSTL(2,1)
          AM2 = BSTL(2,2)
          AN2 = BSTL(2,3)
          AL3 = BSTL(3,1)
          AM3 = BSTL(3,2)


          AN3 = BSTL(3,3)
          EPSCOR = 0.0
          EPSYLD = TENSTF(L,12+ISPG,J)
          IF (EPSSTL .LE. EPSYLD) THEN
              NCRAC = INT(TENSTF(L,ISP,J))
              CALL SPGN (EPSSTL, L, J, TOLER, CHKGUS, PROPER, NSTIFF
     1            , CONVER, IOUT, NCRAC, ISPG, ELNUM, NNEL, IFG)
          ELSE
              TENSTF(L,ISPG+24,J) = 0.10*TENSTF(L,ISPG+6,J)/EPSSTL
              IF (TENSTF(L,18+ISPG,J).LT.EPSYLD .AND. EPSSTL.GE.
     1            EPSYLD) THEN
                  CHKGUS = .TRUE.
                  NSTIFF = 1
                  CONVER = 999.0
                  WRITE (*, *) 'TSTIF 4 999 ELNUM', ELNUM
              ENDIF
              TENSTF(L,15+ISPG,J) = EPSSTL
              TENSTF(L,18+ISPG,J) = EPSSTL
          ENDIF
```

```
                TENXYZ(1,1) = TENSTF(L,24+ISPG,J)
                IMG = IMG + 1
                IF (IMG .EQ. 1) THEN
                    IF (ICOUP .EQ. 1) THEN
                        IF (ISPG .EQ. 1) THEN
                            ISPG1 = ISPG
                            ISPG2 = 2
                            ISPG3 = 3
                        ELSE IF (ISPG .EQ. 2) THEN
                            ISPG1 = ISPG
                            ISPG2 = 3
                            ISPG3 = 1
                        ELSE IF (ISPG .EQ. 3) THEN
                            ISPG1 = ISPG
                            ISPG2 = 1
                            ISPG3 = 2
                        ENDIF
                        IF (TENSTF(L,4,J) .EQ. ISPG2) ISP2 = 1
                        IF (TENSTF(L,5,J) .EQ. ISPG2) ISP2 = 2
                        IF (TENSTF(L,6,J) .EQ. ISPG2) ISP2 = 3
                        IF (TENSTF(L,4,J) .EQ. ISPG3) ISP3 = 1
                        IF (TENSTF(L,5,J) .EQ. ISPG3) ISP3 = 2
                        IF (TENSTF(L,6,J) .EQ. ISPG3) ISP3 = 3
                        EPSTL1 = EPSX*NT(1,1)*NT(1,1) + EPSY*NT(1,2)*NT(1,
     1                      2) + EPSZ*NT(1,3)*NT(1,3) + GAMAXY*NT(1,1)*NT(
     2                      1,2) + GAMAYZ*NT(1,2)*NT(1,3) + GAMAXZ*NT(1,3)
     3                      *NT(1,1)
                        EPSTL2 = EPSX*NT(2,1)*NT(2,1) + EPSY*NT(2,2)*NT(2,
     1                      2) + EPSZ*NT(2,3)*NT(2,3) + GAMAXY*NT(2,1)*NT(
     2                      2,2) + GAMAYZ*NT(2,2)*NT(2,3) + GAMAXZ*NT(2,3)
     3                      *NT(2,1)
                        EPSTL3 = EPSX*NT(3,1)*NT(3,1) + EPSY*NT(3,2)*NT(3,
     1                      2) + EPSZ*NT(3,3)*NT(3,3) + GAMAXY*NT(3,1)*NT(
     2                      3,2) + GAMAYZ*NT(3,2)*NT(3,3) + GAMAXZ*NT(3,3)
     3                      *NT(3,1)
                        NCR(1,1) = HISTO(L,44,J)
                        NCR(1,2) = HISTO(L,45,J)
                        NCR(1,3) = HISTO(L,46,J)
                        NCR(2,1) = HISTO(L,47,J)
                        NCR(2,2) = HISTO(L,48,J)
                        NCR(2,3) = HISTO(L,49,J)
                        NCR(3,1) = HISTO(L,50,J)
                        NCR(3,2) = HISTO(L,51,J)
                        NCR(3,3) = HISTO(L,52,J)
                        DO 370 IMM = 1, 3
                            DO 360 IMP = 1, 3
                                BSTL(IMM,IMP) = 0.0
                                BSTL(IMM,IMP) = BSTL(IMM,IMP) + NT(IMM,1)*
     1                              NCR(IMP,1)
                                BSTL(IMM,IMP) = BSTL(IMM,IMP) + NT(IMM,2)*
     1                              NCR(IMP,2)
                                BSTL(IMM,IMP) = BSTL(IMM,IMP) + NT(IMM,3)*
     1                              NCR(IMP,3)
 360                        CONTINUE
 370                    CONTINUE
                        DO MK = 1, 3
                            DO MN = 1, 3
                                IF (DABS(BSTL(MK,MN)) .LE. 0.01) BSTL(MK,
     1                              MN) = 0.0
                            END DO
                        END DO
                        AL1 = BSTL(1,1)
                        AM1 = BSTL(1,2)
```

```
      AN1 = BSTL(1,3)
      AL2 = BSTL(2,1)
      AM2 = BSTL(2,2)
      AN2 = BSTL(2,3)
      AL3 = BSTL(3,1)
      AM3 = BSTL(3,2)
      AN3 = BSTL(3,3)
      PEPMAX = HISTO(L,41,J)
      PEPMID = HISTO(L,42,J)
      PEPMIN = HISTO(L,43,J)
      PSTL1 = PEPMAX*AL1*AL1 + PEPMID*AM1*AM1 + PEPMIN*
     1    AN1*AN1
      PSTL2 = PEPMAX*AL2*AL2 + PEPMID*AM2*AM2 + PEPMIN*
     1    AN2*AN2
      PSTL3 = PEPMAX*AL3*AL3 + PEPMID*AM3*AM3 + PEPMIN*
     1    AN3*AN3
      IF (TENSTF(L,ISP2,J) .EQ. 0) THEN
          IF (PSTL2.LE.TENSTF(L,9+ISPG,J) .AND. EPSTL2
     1        .GT.TENSTF(L,9+ISPG,J)) THEN
              CHKGUS = .TRUE.
              CONVER = 999.0
              WRITE (*, *) 'TSTIF 5 999 ELNUM', ELNUM
          ENDIF
      ENDIF
C
      IF (TENSTF(L,ISP3,J) .EQ. 0) THEN
          IF (PSTL3.LE.TENSTF(L,9+ISPG,J) .AND. EPSTL3
     1        .GT.TENSTF(L,9+ISPG,J)) THEN
              CHKGUS = .TRUE.
              CONVER = 999.0
              WRITE (*, *) 'TSTIF 6 999 ELNUM', ELNUM
          ENDIF
      ENDIF
      CNST12 = 0.0
      CNST23 = 0.0
      CNST13 = 0.0
      IF (TENSTF(L,ISP,J).EQ.1.0 .AND. TENSTF(L,ISP2,J)
     1    .EQ.1.0) THEN
          CNST12 = DSQRT(TENSTF(L,24+ISPG,J)/HISTO(L,26,
     1        J)*TENSTF(L,24+ISPG2,J)/HISTO(L,26,J))
      ELSE IF (TENSTF(L,ISP,J).EQ.1.0 .AND. TENSTF(L,
     1        ISP2,J).EQ.0.0) THEN
          CNST12 = DSQRT(TENSTF(L,24+ISPG,J)/HISTO(L,26,
     1        J)*HISTO(L,20,J)/HISTO(L,26,J))
          IF(EPSTL2.GT.TENSTF(L,9+ISPG,J))CNST12=0.0
      ELSE IF (TENSTF(L,ISP,J).EQ.0.0 .AND. TENSTF(L,
     1        ISP2,J).EQ.1.0) THEN
          CNST12 = DSQRT(TENSTF(L,24+ISPG2,J)/HISTO(L,26
     1        ,J)*HISTO(L,20,J)/HISTO(L,26,J))
          IF(EPSTL1.GT.TENSTF(L,9+ISPG,J))CNST12=0.0
      ELSE IF (TENSTF(L,ISP,J).EQ.0.0 .AND. TENSTF(L,
     1        ISP2,J).EQ.0.0) THEN
          CNST12 = DSQRT(HISTO(L,20,J)/HISTO(L,26,J)*
     1        HISTO(L,20,J)/HISTO(L,26,J))
          IF (EPSTL2.GT.TENSTF(L,9+ISPG,J) .OR. EPSTL1
     1        .GT.TENSTF(L,9+ISPG,J)) CNST12 = 0.0
      ENDIF
      IF (TENSTF(L,ISP,J).EQ.1.0 .AND. TENSTF(L,ISP3,J)
     1    .EQ.1.0) THEN
          CNST13 = DSQRT(TENSTF(L,24+ISPG,J)/HISTO(L,26,
     1        J)*TENSTF(L,24+ISPG3,J)/HISTO(L,26,J))
      ELSE IF (TENSTF(L,ISP,J).EQ.1.0 .AND. TENSTF(L,
     1        ISP3,J).EQ.0.0) THEN
```

```
                    CNST13 = DSQRT(TENSTF(L,24+ISPG,J)/HISTO(L,26,
     1                  J)*HISTO(L,20,J)/HISTO(L,26,J))
                    IF(EPSTL3.GT.TENSTF(L,9+ISPG,J))CNST13=0.0
                ELSE IF (TENSTF(L,ISP,J).EQ.0.0 .AND. TENSTF(L,
     1                  ISP3,J).EQ.1.0) THEN
                    CNST13 = DSQRT(TENSTF(L,24+ISPG3,J)/HISTO(L,26
     1                  ,J)*HISTO(L,20,J)/HISTO(L,26,J))
                    IF(EPSTL1.GT.TENSTF(L,9+ISPG,J))CNST13=0.0
                ELSE IF (TENSTF(L,ISP,J).EQ.0.0 .AND. TENSTF(L,
     1                  ISP3,J).EQ.0.0) THEN
                    CNST13 = DSQRT(HISTO(L,20,J)/HISTO(L,26,J)*
     1                  HISTO(L,20,J)/HISTO(L,26,J))
                    IF (EPSTL3.GT.TENSTF(L,9+ISPG,J) .OR. EPSTL1
     1                  .GT.TENSTF(L,9+ISPG,J)) CNST13 = 0.0
                ENDIF
                IF (TENSTF(L,ISP3,J).EQ.1.0 .AND. TENSTF(L,ISP2,J)
     1              .EQ.1.0) THEN
                    CNST23 = DSQRT(TENSTF(L,24+ISPG3,J)/HISTO(L,26
     1                  ,J)*TENSTF(L,24+ISPG2,J)/HISTO(L,26,J))
                ELSE IF (TENSTF(L,ISP3,J).EQ.1.0 .AND. TENSTF(L,
     1                  ISP2,J).EQ.0.0) THEN
                    CNST23 = DSQRT(TENSTF(L,24+ISPG3,J)/HISTO(L,26
     1                  ,J)*HISTO(L,20,J)/HISTO(L,26,J))
                    IF(EPSTL2.GT.TENSTF(L,9+ISPG,J))CNST23=0.0
                ELSE IF (TENSTF(L,ISP3,J).EQ.0.0 .AND. TENSTF(L,
     1                  ISP2,J).EQ.1.0) THEN
                    CNST23 = DSQRT(TENSTF(L,24+ISPG2,J)/HISTO(L,26
     1                  ,J)*HISTO(L,20,J)/HISTO(L,26,J))
                    IF(EPSTL3.GT.TENSTF(L,9+ISPG,J))CNST23=0.0
                ELSE IF (TENSTF(L,ISP2,J).EQ.0.0 .AND. TENSTF(L,
     1                  ISP3,J).EQ.0.0) THEN
                    CNST23 = DSQRT(HISTO(L,20,J)/HISTO(L,26,J)*
     1                  HISTO(L,20,J)/HISTO(L,26,J))
                    IF (EPSTL3.GT.TENSTF(L,9+ISPG,J) .OR. EPSTL2
     1                  .GT.TENSTF(L,9+ISPG,J)) CNST23 = 0.0
                ENDIF
                TEMP11(2,1) = CNST12*HISTO(L,26,J)*HISTO(L,27,J)
                TEMP11(1,2) = CNST12*HISTO(L,26,J)*HISTO(L,27,J)
                TEMP11(3,1) = CNST13*HISTO(L,26,J)*HISTO(L,27,J)
                TEMP11(1,3) = CNST13*HISTO(L,26,J)*HISTO(L,27,J)
                TEMP11(2,3) = CNST23*HISTO(L,26,J)*HISTO(L,27,J)
                TEMP11(3,2) = CNST23*HISTO(L,26,J)*HISTO(L,27,J)
                CALL DTRANS (TEMP11, NT, IOUT)
                ICOUP = 1
            ENDIF
          ENDIF
          CALL DTRANS (TENXYZ, NT, IOUT)
          DO 410 IJ = 1, 6
            DO 400 JK = 1, 6
                DTSTIF(IJ,JK) = TENXYZ(IJ,JK) + DTSTIF(IJ,JK)
 400        CONTINUE
 410      CONTINUE
        ENDIF
 420  CONTINUE
      RETURN
      END

C ----------------------------- A R A N G E --------------------
C     INCLUDE(PROCESS)
      SUBROUTINE ARANGE(ICRACK,NCRACK,IG,IL)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
```

```
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C************************************************************
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
C
C
C               SUBROUTINE TO ELIMINATE INFORMATION REGARDING A VERY
C               STIFF CRACK WHICH IS COMPLETELY CLOSING. THE ARRAY
C               'CRACK' WILL BE REARRANGED BY POPPING UP THE DATA
C               FOR THE REMAINING CRACKS.
C
C
      CRACK(IG,1,IL) = CRACK(IG,1,IL) - 1.0
      DO 100 I = ICRACK, NCRACK - 1
C
C                CRACK INITIATION VALUE
C
          CRACK(IG,37+I,IL) = CRACK(IG,38+I,IL)
C
C                CRACK DIRECTION
C
          CRACK(IG,137+I,IL) = CRACK(IG,146+I,IL)
          CRACK(IG,138+I,IL) = CRACK(IG,147+I,IL)
          CRACK(IG,139+I,IL) = CRACK(IG,148+I,IL)
          CRACK(IG,140+I,IL) = CRACK(IG,149+I,IL)
          CRACK(IG,141+I,IL) = CRACK(IG,150+I,IL)
          CRACK(IG,142+I,IL) = CRACK(IG,151+I,IL)
          CRACK(IG,143+I,IL) = CRACK(IG,152+I,IL)
          CRACK(IG,144+I,IL) = CRACK(IG,153+I,IL)
          CRACK(IG,145+I,IL) = CRACK(IG,154+I,IL)
C
C                OLD STRAIN OF THE CRACK
C
          CRACK(IG,47+I,IL) = CRACK(IG,48+I,IL)
C
C               CURRENT STRAIN OF THE CRACK
C
          CRACK(IG,57+I,IL) = CRACK(IG,58+I,IL)
C
C               DIRECT STIFFNESS OF THE CRACK
C
          CRACK(IG,87+I,IL) = CRACK(IG,88+I,IL)
C
C               SHEAR STIFFNESS OF THE CRACK
C
          CRACK(IG,97+I,IL) = CRACK(IG,98+I,IL)
          CRACK(IG,107+I,IL) = CRACK(IG,108+I,IL)
C
C               STATE OF THE CRACK
C
          CRACK(IG,117+I,IL) = CRACK(IG,118+I,IL)
C
C               SHEAR STRAIN OF THE CRACK
C
          CRACK(IG,67+I,IL) = CRACK(IG,68+I,IL)
          CRACK(IG,77+I,IL) = CRACK(IG,78+I,IL)
C
C               TERMINATION STRAIN FOR THE CRACK
```

```
C
         CRACK(IG,127+I,IL) = CRACK(IG,128+I,IL)
  100 CONTINUE
      RETURN
      END
C
C
C
C ----------------------------- C R A C K E --------------------
C
C     INCLUDE(PROCESS)
      SUBROUTINE CRACKE(EPSCRK,PROPER,ES,ICRACK,IGAUSS,ILAYER)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C************************************************************
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      DIMENSION PROPER(25)
C
C
C              SUBROUTINE TO DETERMINE SECANT YOUNG'S MODULUS CORRES-
C              PONDING TO THE RECENT STRAIN ACROSS THE CRACK, 'EPSCRK'.
C
C
C               RETRIEVE THE CRACK INITIATION STRESS, FT, AND THE
C               CRACK TERMINATION STRAIN, EPSNOT .
C
C
      FT = CRACK(IGAUSS,37+ICRACK,ILAYER)
      EPSNOT = CRACK(IGAUSS,127+ICRACK,ILAYER)
C
C               FOR CRACK TREATMENT RELATED TO TENSION-STIFFENING ...
C
      IF (PROPER(16) .GT. 0.0) GO TO 100
C
C               CRACKING RELATED TO MATERIAL SOFTENING MODEL
C
C
C
C               TRANST = STRAIN AT THE TRANSITION POINT FOR BILINEAR
C                        STRAIN SOFTENING MODEL.
C
      TRANST = EPSNOT*2./9.
C
C         ..... DETERMINE STEP FUNCTIONS FOR BILINEAR STRAIN SOFTENING.
C
      S1 = 0.0
      S2 = 0.0
      IF (EPSCRK.GT.0. .AND. EPSCRK.LE.TRANST) S1 = 1.0
      IF (EPSCRK.GT.TRANST .AND. EPSCRK.LT.EPSNOT) S2 = 1.
      ES = 3.*FT*(S1*(-7.+(7.*EPSNOT)/(3.*EPSCRK))+S2*(-1.+EPSNOT/EPSCRK
     1    ))/(7.*EPSNOT)
      RETURN
C
C                 **     TENSION STIFFENING CRACK   **
C
```

```
  100 CONTINUE
      S = 0.0
      IF (EPSCRK.GT.0.0 .AND. EPSCRK.LE.EPSNOT) S = 1.0
      ES = S*(FT*(1./EPSCRK-1./EPSNOT))
      RETURN
      END
C
C
C
C
C ----------------------------  I N I T E S  --------------------
C
C     INCLUDE(PROCESS)
      SUBROUTINE INITES(SIGCRK,PROPER,ES,ICRACK,IGAUSS,ILAYER,IOUT)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C*************************************************************
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      DIMENSION PROPER(25)
C
C
C              SUBROUTINE TO DETERMINE SECANT YOUNG'S MODULUS CORRES-
C              PONDING TO THE RECENT STRESS ACROSS THE CRACK, 'SIGCRK'.
C
C
C               RETRIEVE THE CRACK INITIATION STRESS, FT, AND
C               THE CRACK TERMINATION STRAIN, EPSNOT .
C
C
      FT = CRACK(IGAUSS,37+ICRACK,ILAYER)
      EPSNOT = CRACK(IGAUSS,127+ICRACK,ILAYER)
C
C               FOR CRACK TREATMENT RELATED TO TENSION-STIFFENING ...
C
      IF (PROPER(16) .GT. 0.0) GO TO 100
C
C               CRACKING RELATED TO MATERIAL SOFTENING MODEL
C
C         ..... DETERMINE STEP FUNCTIONS FOR BILINEAR STRAIN SOFTENING.
C
      S1 = 0.0
      S2 = 0.0
      IF (SIGCRK.LT.FT .AND. SIGCRK.GE.FT/3.0) S1 = 1.0
      IF (SIGCRK.GE.0.0 .AND. SIGCRK.LT.FT/3.0) S2 = 1.0
      ES = (3.*FT*SIGCRK/EPSNOT)*(S1/(FT-SIGCRK)+S2/(3.*FT-7.*SIGCRK))
      RETURN
C
C                 **     TENSION STIFFENING CRACK   **
C
  100 CONTINUE
      S = 0.0
      IF (SIGCRK.GE.0.0 .AND. SIGCRK.LT.FT) S = 1.0
      ES = S*FT*SIGCRK/(EPSNOT*(FT-SIGCRK))
      RETURN
      END
```

```
C
C
C
C ---------------------------- C R K C O N -----------------------
C
C     INCLUDE(PROCESS)
      SUBROUTINE CRKCON(PROPER,DCRACK,ILAYER,ICRACK,IGAUSS,EPSCRK,ES,
     $         CHKGUS,NSTIFF,CONVER,TOLER,IOUT,EPSOLD,IFLAG)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C CRKSTN
C
C*************************************************************
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      DIMENSION PROPER(25)
      LOGICAL CHKGUS
C
C                THE CRACK HAD BEEN OVER-SOFTENED IN THE PREVIOUS ITER..
C
      IF (DCRACK .LT. ES) DIFFER = (ES-DCRACK)*100./ES
      AREA = 0.0
C
C
C     ......... DETERMINE THE STRAIN CORRESPONDING TO CRACK STIFFNESS
C                'DCRACK' EMPLOYED IN THE CURRENT ITERATION.
C
C
      CALL CRKSTN (PROPER, DCRACK, EPSOLD, ILAYER, ICRACK, IGAUSS, IOUT)
      SIGOLD = DCRACK*EPSOLD
      SIGNEW = ES*EPSCRK
      FT = CRACK(IGAUSS,37+ICRACK,ILAYER)
      EPSNOT = CRACK(IGAUSS,127+ICRACK,ILAYER)
C
C                AREA = AREA UNDER THE STRESS-STRAIN DIAGRAM OF THE
C                       CURRENT CRACK INDICATING THE AMOUNT OF ENERGY
C                       TO BE CONSUMED FOR COMPLETE OPENING OF THE CRK.
C
C     .............. MATERIAL SOFTENING CRACK
C
C
      IF (PROPER(16) .LT. 0.0) AREA = 5.0*FT*EPSNOT/18.0
C
C           .... CRACK RELATED TO TENSION STIFFENING
C
      IF (PROPER(16) .GT. 0.0) AREA = .50*FT*EPSNOT
C
C
C                COMPARE THE DIFFERENCE IN ENERGY ABSORPTION, DETERMINED
C                BY THE AREA ENCLOSED BY THE EMPLOYED AND THE UPDATED
C                CRACK STIFFNESS, WITH THE TOTAL ENERGY OR AREA UNDER
C                THE STRESS-STRAIN CURVE OF THE CURRENT CRACK.
C
C
      SUBARA = .50*FT*(EPSCRK-EPSOLD)
      IF (PROPER(16) .LT. 0.0) THEN
         IF (SIGOLD.GE.FT/3.0 .AND. SIGNEW.LE.FT/3.) SUBARA = AREA -
     1        .50*(FT*EPSOLD+SIGNEW*EPSNOT)
```

```
         IF (SIGOLD.LE.FT/3.0 .AND. SIGNEW.LE.FT/3.0) SUBARA = .50*
     1         EPSNOT*(SIGOLD-SIGNEW)
      ENDIF
C
      DIFFER = 100.0*(SUBARA/AREA)
      IF (DIFFER .LE. TOLER) RETURN
      IF (DIFFER .GT. TOLER) THEN
C
C                 UPDATE THE CRACK STIFFNESS
C
         CRACK(IGAUSS,87+ICRACK,ILAYER) = ES
         IFLAG = 1
         CHKGUS = .TRUE.
         NSTIFF = 1
         CONVER = 999.00
         WRITE (*, *) 'CRKCON 999 ELNUM', 'CRACK#', ICRACK
      ENDIF
      RETURN
      END
C
C
C
C _____________________________ C  R  K  S  T  N  ____________________
C
C     INCLUDE(PROCESS)
      SUBROUTINE CRKSTN(PROPER,DCRACK,EPDCRK,ILAYER,ICRACK,IGAUSS,IOUT)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C***********************************************************
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      DIMENSION PROPER(25)
C
C
C                 SUBROUTINE TO DETERMINE THE STRAIN ACROSS A CRACK FOR
C                 A GIVEN CRACK STIFFNESS.
C
C
      FT = CRACK(IGAUSS,37+ICRACK,ILAYER)
      EPSNOT = CRACK(IGAUSS,127+ICRACK,ILAYER)
C
C                 FOR TENSION STIFFENING CRACK
C
      IF (PROPER(16) .GT. 0.0) GO TO 100
C
C            ....... CRACKING RELATED TO MATERIAL STRAIN SOFTENING
C
C
C                 DETERMINE STEP FUNCTIONS FOR BILINEAR STRAIN SOFTENING
C
      S1 = 0.0
      S2 = 0.0
C
C                 ETRANS = CRACK STIFFNESS AT THE BILINEAR TRANSITION PNT
C
      ETRANS = 1.50*FT/EPSNOT
```

```
      IF (DCRACK .GE. ETRANS) S1 = 1.0
      IF (DCRACK .LT. ETRANS) S2 = 1.0
      EPDCRK = FT*EPSNOT*(S1/(3.0*FT+DCRACK*EPSNOT)+3.*S2/(3.*FT+7.*
     1    DCRACK*EPSNOT))
      RETURN
C
C                    FOR TENSION STIFFENING CRACK
C
  100 CONTINUE
      EPDCRK = FT*EPSNOT/(DCRACK*EPSNOT+FT)
      RETURN
      END
C
C
C
C ____________________________ O  V  E  R  F  T  ____________________
C
C     INCLUDE(PROCESS)
      SUBROUTINE OVERFT(EPSOLD,TOLER,IGAUSS,ICRACK,ILAYER,PROPER,DCRACK,
     $                 CHKGUS,NSTIFF,CONVER,IOUT,SIGCRK,ES,IELEM)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C CRACKE
C
C***************************************************************
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      DIMENSION PROPER(25)
      LOGICAL CHKGUS
C
C
C              SUBROUTINE TO CORRECT OVERSHOOTING BEYOND THE CRACKING
C              STRESS FOR THE CURRENT CRACK. EVEN THOUGH THE CALCULATE
C              'E' BASED ON THE CURRENT STRAIN ACROSS THE CRACK DID NO
C              SHOW THE NEED FOR UPDATING THE CRACK STIFFNESS THE ENER
C              GY CRITERION FOR SOFTENING THE CRACK AND HENCE ELIMINAT
C              ING THE OVERSHOOTING MUST BE USED.
C
C
C            .... GET THE INITIATION STRESS AND THE TERMINATION STRAIN.
C
      FT = CRACK(IGAUSS,37+ICRACK,ILAYER)
      EPSNOT = CRACK(IGAUSS,127+ICRACK,ILAYER)
      IF (SIGCRK.GT.FT .AND. ES.GT.DCRACK) THEN
C
C              EVEN THOUGH THERE IS OVERSHOOTING THE UPDATED SECANT
C              STIFFNESS BECAME LARGER THAN ITS PREVIOUS VALUE 3
C
         WRITE(*,900)ICRACK,IGAUSS,ILAYER,SIGCRK,FT,DCRACK,ES,IELEM
         STOP
      ENDIF
C
C
C              DETERMINE A NEW STRAIN ON THE STRESS-STRAIN CURVE
C              OF THE CRACK. THE AREA ENCLOSED BY THE OLD AND THE
C              NEW CRACK STRAIN IS EQUAL TO THE TOLERANCE.
C
C
```

```
C          .... FOR TENSION STIFFENING CRACK
C
      IF (PROPER(16) .GT. 0.0) EPSNEW = EPSOLD + .010*TOLER*EPSNOT
      IF (PROPER(16).LT.0.0 .AND. EPSOLD.LT.2.*EPSNOT/9.) THEN
          EPSNEW = EPSOLD + .010*TOLER*5.0*EPSNOT/9.0
          IF (EPSNEW .GT. 2.*EPSNOT/9.) EPSNEW = 2.*EPSNOT/9.0
      ENDIF
C
C
C               WITH THE NEW CRACK STRAIN CALCULATE ITS CORRESPONDING
C               SECANT 'E'.
C
      CALL CRACKE (EPSNEW, PROPER, ESNEW, ICRACK, IGAUSS, ILAYER, IOUT)
      EINITL = PROPER(1)
      ESNEW = DMAX1(EINITL/1000.0,ESNEW)
C
C               UPDATE THE CRACK STIFFNESS
C
      CRACK(IGAUSS,87+ICRACK,ILAYER) = ESNEW
      CHKGUS = .TRUE.
      NSTIFF = 1
      CONVER = 999.
      RETURN
  900   FORMAT(1X,'     ERROR IN SUBROUTINE CRKPNT',/,5X,
     1          'FOR CRACK NO.',I3,' LOCATED IN GAUSS POINT',
     2           I2,' OF LAYER',I3,' IN THE CURRENT ELEMENT',
     3           /,5X,'CRACK STRESS=',D12.5,'  INITIATION',
     4           ' STRESS VALUE=',D12.5,/,5X,'EMPLOYED CRACK',
     5           ' STIFFNESS=',D12.5,' <  UPDATED CRACK ',
     6           'STIFFNESS=',D12.5,' 3','ELNUM',I2)
      END
C
C *******************************************************************
C
C     INCLUDE(PROCESS)
        SUBROUTINE SPGN(EPSN,IGAUSS,ILAYER,TOLER,CHKGUS,PROPER
     $     ,NSTIFF,CONVER,IOUT,NCRAC,ICRACK,ELNUM,NNEL,IFG)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C CKTEN     SPCON     LYINFO    GTLTK     GETTHK    OVRFT
C
C************************************************************
      INTEGER ELNUM
      REAL*8 LTHICK
C
C
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON/ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      DIMENSION LTHICK(9),ZS(9),DCS(3,3)
      REAL*8 PROPER(25)
      LOGICAL CHKGUS
      IF (NCRAC .EQ. 1) THEN
C
C       RECOVER THE OLD STRAINS,STIFFNESS,STATE FOR THE CURRENT CRACK
C
          EPSO = TENSTF(IGAUSS,15+ICRACK,ILAYER)
          STATE = TENSTF(IGAUSS,21+ICRACK,ILAYER)
```

```
          SPRING = TENSTF(IGAUSS,24+ICRACK,ILAYER)
          SIGMT = SPRING*EPSN
          EINITL = PROPER(1)
          CKSTIN = TENSTF(IGAUSS,9+ICRACK,ILAYER)
C
C       CLOSED SPRING OPENS(CRACK OPENING)
C
          IF (EPSN.GT.0.0 .AND. STATE.EQ.4.0) THEN
              EST = TENSTF(IGAUSS,27+ICRACK,ILAYER)
C ----- IF STIFFNESS LESS THEN 10000TH THAT OF CONCRETE SET TO THAT!
              IF (EST .LT. EINITL/1000.) EST = EINITL/1000.0
              TENSTF(IGAUSS,24+ICRACK,ILAYER) = EST
              TENSTF(IGAUSS,18+ICRACK,ILAYER) = CKSTIN
              TENSTF(IGAUSS,21+ICRACK,ILAYER) = 3.0
              CHKGUS = .TRUE.
              NSTIFF = 1
              CONVER = 999.0
              WRITE (*, *) 'SPGN 1 999 ELNUM', ELNUM
              GO TO 100
          ENDIF
C
C        UNLOADING AND RELOADING CRACKS SET TO LOADING CRACKS
C
          IF (EPSN.GT.0.0 .AND. EPSN.GE.EPSO) THEN
              IF (STATE.EQ.2.0 .OR. STATE.EQ.3.0) TENSTF(IGAUSS,21+
     1            ICRACK,ILAYER) = 1.0
              TENSTF(IGAUSS,15+ICRACK,ILAYER) = EPSN
              TENSTF(IGAUSS,18+ICRACK,ILAYER) = EPSN
C ----- IF STIFFNESS LESS THEN 10000TH THAT OF CONCRETE SET TO THAT!
              IF (DABS(SPRING) .LE. EINITL/1000.0) RETURN
              CALL CKTEN(EPSN,EST,IGAUSS,ICRACK,ILAYER,ELNUM,NNEL,IFG)
C ----- IF STIFFNESS LESS THEN 10000TH THAT OF CONCRETE SET TO THAT!
              IF (DABS(EST) .LE. EINITL/1000.0) THEN
                  TENSTF(IGAUSS,24+ICRACK,ILAYER) = EINITL/1000.0
                  NSTIFF = 1
                  CONVER = 999.0
                  WRITE (*, *) 'SPGN 2 999 ELNUM', ELNUM
                  GO TO 100
              ENDIF
C
C      CHECK CONVERGENCE OF THE STIFFNESS
C
              IFLAG = 0
              FT = TENSTF(IGAUSS,6+ICRACK,ILAYER)
              CALL SPCON (SPRING, ILAYER, ICRACK, IGAUSS, EPSN, EST,
     1            CHKGUS, NSTIFF, CONVER, TOLER, IOUT, EPSO, IFLAG,
     2            ELNUM, NNEL, IFG)
              IF (CONVER .EQ. 999) WRITE (*, *) 'SPGN 3 999 ELNUM',
     1            ELNUM
              IF (SIGMT.LT.FT .AND. IFG.EQ.1) GO TO 100
              IF (SIGMT.GE.FT .AND. IFLAG.EQ.0 .AND. IFG.EQ.1 .OR. IFG
     1            .EQ.2 .AND. IFLAG.EQ.0) THEN
                  IF (IFG .EQ. 2) THEN
                      EPSY = TENSTF(IGAUSS,12+ICRACK,ILAYER)
                      SIGYLD = TENSTF(IGAUSS,ICRACK+63,ILAYER)
                      SRATIO = TENSTF(IGAUSS,ICRACK+60,ILAYER)
                      ILL = TENSTF(IGAUSS,ICRACK+66,ILAYER)
                      CALL LYINFO (ILL, LTHICK, ZS, MATRL, DCS, NNEL,
     1                    ELNUM, NIPXI, NIPETA, NIPSI)
                      CALL GTLTK(L,ELNUM,NNEL,THICKL,ZSI,LTHICK,ZS)
                      CALL GETTHK (L, ELNUM, NNEL, THICKE, RAD)
                      RHO = THICKL/THICKE
C     WRITE(*,*)'OVRFT SIGYLD',SIGYLD,'SRATIO',SRATIO,'RHO',RHO,'LY',ILL
```

```
                    EPSCR1 = SRATIO*FT*EPSY/SIGYLD
                    EPSCR2 = (SRATIO*FT/SIGYLD)*(1.0+(1.0+RHO*SRATIO)/
     1                  (10.0*RHO*SRATIO))*EPSY
                    EPSCR3 = (1.0-(1.0/(2.*RHO*SRATIO))*(SRATIO*FT/
     1                  SIGYLD))*EPSY
                    EPSCR4 = EPSY
C     WRITE(*,*)'EP1',EPSCR1,'EP2',EPSCR2,'EP3',EPSCR3,'EP4',EPSCR4
                    C = 550.0
                    FT1 = FT
                    FT2 = (1.0-(1.0+RHO*SRATIO)/10.0)*FT*DEXP((-C*(
     1                  EPSCR2-EPSCR1)))
                    FT3 = FT*DEXP((-C*(EPSCR3-EPSCR1)))/2.0
                    FT4 = FT/10.0
                    IF (EPSO .LE. EPSCR2) THEN
                        IF (SIGMT .LE. FT1) GO TO 100
                    ELSE IF (EPSO.GT.EPSCR2 .AND. EPSO.LE.EPSCR3) THEN
                        IF (SIGMT .LE. FT2) GO TO 100
                    ELSE IF (EPSO.GT.EPSCR3 .AND. EPSO.LE.EPSCR4) THEN
                        IF (SIGMT .LE. FT3) GO TO 100
                    ENDIF
                ENDIF
                CALL OVRFT (EPSO, TOLER, IGAUSS, ICRACK, ILAYER,
     1              EININTL, SPRING, CHKGUS, NSTIFF, CONVER, IOUT,
     2              SIGMT, EST, ELNUM, NNEL, IFG)
                GO TO 100
            ENDIF
        ENDIF
C
C     PARTIAL OPENING OR CLOSING OF CRACK
C
        IF (EPSN.GT.0.0 .AND. EPSN.LT.EPSO) THEN
            PEPSN = TENSTF(IGAUSS,18+ICRACK,ILAYER)
            IF (EPSN .LT. PEPSN) THEN
                TENSTF(IGAUSS,21+ICRACK,ILAYER) = 2.0
            ELSE
                TENSTF(IGAUSS,21+ICRACK,ILAYER) = 3.0
            ENDIF
            TENSTF(IGAUSS,18+ICRACK,ILAYER) = EPSN
            GO TO 100
        ENDIF
C
C    COMPLETE CLOSING OF CRACK
C
        IF (EPSN.LT.0.0 .AND. STATE.NE.4.0) THEN
            TENSTF(IGAUSS,21+ICRACK,ILAYER) = 4.0
            TENSTF(IGAUSS,27+ICRACK,ILAYER) = TENSTF(IGAUSS,24+ICRACK,
     1          ILAYER)
C ----- IF STIFFNESS LESS THEN 10000TH THAT OF CONCRETE SET TO THAT!
            TENSTF(IGAUSS,24+ICRACK,ILAYER) = PROPER(1)/1000.0
        ENDIF
      ENDIF
  100 CONTINUE
      RETURN
      END
C************************************************************
C
C     INCLUDE(PROCESS)
        SUBROUTINE OVRFT(EPSO,TOLER,IGAUSS,I,ILAYER,EININTL,
     $                          SPRING,CHKGUS,NSTIFF,CONVER,IOUT,
     $                          SIGMT,EST,ELNUM,NNEL,IFG)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
```

```
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C LYINFO    GTLTK     GETTHK    CKTEN
C
C***********************************************************
C
      INTEGER ELNUM
      REAL*8 LTHICK
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON/ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      DIMENSION LTHICK(9),ZS(9),DCS(3,3)
        LOGICAL CHKGUS
C
C       SUBROUTINE CORRECTS OVERSHOOTING OF STRESS IN SPRING1 DIRECTION
C
      FT = TENSTF(IGAUSS,I+6,ILAYER)
      CKSTIN = TENSTF(IGAUSS,9+I,ILAYER)


      EPSY = TENSTF(IGAUSS,12+I,ILAYER)
C
      IF (SIGMT.GT.1.10*FT .AND. DABS(EST).GT.DABS(SPRING)) THEN
          WRITE (*, *) 'EST', EST, 'SPRING', SPRING, 'ELNUM', ELNUM
          WRITE (*, *) 'ILAYER', ILAYER, 'IGAUSS', IGAUSS
          WRITE (*, 900)
          STOP
      ENDIF
C
C       IFG=2
      IF (IFG .EQ. 1) THEN
          FT1 = 0.35*FT
          FT2 = 0.10*FT
          EPSCR = CKSTIN
          EPSCR1 = EPSY*0.50
          EPSCR2 = EPSY
          AREA = 0.5*(EPSCR1-EPSCR)*(FT+FT1) + 0.5*(EPSCR2-EPSCR1)*(FT2+
     1        FT1)
          EPSTN = EPSO + TOLER*AREA/(FT*0.5*300.0)
      ELSE IF (IFG .EQ. 2) THEN
          SIGYLD = TENSTF(IGAUSS,I+63,ILAYER)
          SRATIO = TENSTF(IGAUSS,I+60,ILAYER)
          ILL = TENSTF(IGAUSS,I+66,ILAYER)
          CALL LYINFO (ILL, LTHICK, ZS, MATRL, DCS, NNEL, ELNUM, NIPXI,
     1        NIPETA, NIPSI)
          CALL GTLTK (L, ELNUM, NNEL, THICKL, ZSI, LTHICK, ZS)
          CALL GETTHK (L, ELNUM, NNEL, THICKE, RAD)
          RHO = THICKL/THICKE
C     WRITE(*,*)'OVRFT SIGYLD',SIGYLD,'SRATIO',SRATIO,'RHO',RHO,'LY',ILL
          EPSCR1 = SRATIO*FT*EPSY/SIGYLD
          EPSCR2 = (SRATIO*FT/SIGYLD)*(1.0+(1.0+RHO*SRATIO)/(10.0*RHO*
     1        SRATIO))*EPSY
          EPSCR3 = (1.0-(1.0/(2.*RHO*SRATIO))*(SRATIO*FT/SIGYLD))*EPSY
          EPSCR4 = EPSY
C     WRITE(*,*)'EPSCR1',EPSCR1,'EPSCR2',EPSCR2,'EPSCR3',EPSCR3
C     WRITE(*,*)'EPSCR4',EPSCR4
C
          C = 550.0
          FT1 = FT
          FT2=(1.0-(1.0+RHO*SRATIO)/10.0)*FT*DEXP((-C*(EPSCR2-EPSCR1)))
          FT3 = FT*DEXP((-C*(EPSCR3-EPSCR1)))/2.0
          FT4 = FT/10.0
```

```
C     WRITE(*,*)'FT1',FT1,'FT2',FT2,'FT3',FT3,'FT4',FT4
          AREA = 0.5*(FT1-FT2)*(EPSCR2-EPSCR1) + (EPSCR2-EPSCR1)*FT2 +
     1        0.5*(FT2-FT3)*(EPSCR3-EPSCR2) + (EPSCR3-EPSCR2)*FT3 + 0.5*
     2        (FT3-FT4)*(EPSCR4-EPSCR3) + (EPSCR4-EPSCR3)*FT4
          IF (EPSO .LT. EPSCR2) THEN
              EPSTN = EPSO + TOLER*AREA/(FT1*0.5*100.0)
              EPSTN = DMIN1(EPSCR2,EPSTN)
          ELSE IF (EPSO.GE.EPSCR2 .AND. EPSO.LE.EPSCR3) THEN
              EPSTN = EPSO + TOLER*AREA/(FT2*0.5*100.0)
              EPSTN = DMIN1(EPSCR3,EPSTN)
          ELSE IF (EPSO.GE.EPSCR3 .AND. EPSO.LE.EPSCR4) THEN
              EPSTN = EPSO + TOLER*AREA/(FT3*0.5*100.0)
              EPSTN = DMIN1(EPSCR4,EPSTN)
          ENDIF
      ENDIF
C
C      FIND NEW STIFFNESS
C
      CALL CKTEN (EPSTN, ESTN, IGAUSS, I, ILAYER, ELNUM, NNEL, IFG)
C
C ----- IF STIFFNESS LESS THEN 10000TH THAT OF CONCRETE SET TO THAT!
      IF (DABS(ESTN) .LE. EINITL/1000.0) ESTN = EINITL/1000.0
      TENSTF(IGAUSS,24+I,ILAYER) = ESTN
      TENSTF(IGAUSS,15+I,ILAYER) = EPSTN
      TENSTF(IGAUSS,18+I,ILAYER) = EPSTN
      CONVER = 999.0
      NSTIFF = 1
      CHKGUS = .TRUE.
      RETURN
  900          FORMAT(1X,'   ERROR IN SPG.OVERSHOOT1   HIGHER STIFFNESS
     1                    ENCOUNTERD')
      END
C
C************************************************************
C
C     INCLUDE(PROCESS)
      SUBROUTINE SPCON(SPRING,ILAYER,I,IGAUSS,EPSN,EST,
     $                CHKGUS,NSTIFF,CONVER,TOLER,IOUT,EPSO,
     $                IFLAG,ELNUM,NNEL,IFG)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C SPSTN     LYINFO    GTLTK     GETTHK
C
C***********************************************************
      REAL*8 LTHICK
      INTEGER ELNUM
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON/ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      DIMENSION LTHICK(9),ZS(9),DCS(3,3)
      LOGICAL CHKGUS
      IF (SPRING .LT. EST) THEN
C
C CRACK SPRING HAS BEEN OVERSOFTENED IN PREVIOUS ITERATION
C
          DIFFER = (EST-SPRING)*100/EST
          WRITE (*, 900) I, IGAUSS, ILAYER, SPRING, EST, DIFFER
C               END IF
```

```
      ENDIF
C
C     FIND STRAIN CORRESPONDING TO SPRING
C
      FT = TENSTF(IGAUSS,I+6,ILAYER)
      CKSTIN = TENSTF(IGAUSS,I+9,ILAYER)
      EPSY = TENSTF(IGAUSS,12+I,ILAYER)
C
      IF (SPRING .GT. EST) THEN
          CALL SPSTN(SPRING,EPSO,ILAYER,I,IGAUSS,EPSN,ELNUM,NNEL,IFG)
C
          AREA = 0
          IF (IFG .EQ. 1) THEN
              FT1 = 0.35*FT
              FT2 = 0.10*FT
              EPSCR = CKSTIN
              EPSCR1 = EPSY*0.50
              EPSCR2 = EPSY
              IF (EPSN .LE. EPSCR1) THEN
                  SUBARA = 0.5*FT*(EPSN-EPSO)
              ELSE IF (EPSN.GE.EPSCR1 .AND. EPSO.LE.EPSCR1) THEN
                  SUBARA = 0.50*(EPSCR1-EPSO)*FT + 0.50*FT1*(EPSCR1-
     1                EPSCR) + 0.50*(EPSN-EPSCR1)*(FT1+EST*EPSN) - 0.5*(
     2                EPSN-EPSCR)*EST*EPSN
              ELSE
                  SUBARA = 0.5*(EPSO-EPSCR)*SPRING*EPSO + 0.50*(EPSN-
     1                EPSO)*(EPSO*SPRING+EPSN*EST) - 0.50*(EPSN-EPSCR)*
     2                EPSN*EST
              ENDIF
              AREA = 0.5*(EPSCR1-EPSCR)*(FT+FT1) + 0.5*(EPSCR2-EPSCR1)*(
     1            FT2+FT1)
C
          ELSE IF (IFG .EQ. 2) THEN
              SIGYLD = TENSTF(IGAUSS,I+63,ILAYER)
              SRATIO = TENSTF(IGAUSS,I+60,ILAYER)
              ILL = TENSTF(IGAUSS,I+66,ILAYER)
              CALL LYINFO (ILL, LTHICK, ZS, MATRL, DCS, NNEL, ELNUM,
     1            NIPXI, NIPETA, NIPSI)
              CALL GTLTK (L, ELNUM, NNEL, THICKL, ZSI, LTHICK, ZS)
              CALL GETTHK (L, ELNUM, NNEL, THICKE, RAD)
              RHO = THICKL/THICKE
              EPSCR1 = SRATIO*FT*EPSY/SIGYLD
              EPSCR2 = (SRATIO*FT/SIGYLD)*(1.0+(1.0+RHO*SRATIO)/(10.0*
     1            RHO*SRATIO))*EPSY
              EPSCR3 = (1.0-(1.0/(2.*RHO*SRATIO))*(SRATIO*FT/SIGYLD))*
     1            EPSY
              EPSCR4 = EPSY
              C = 550.0
              FT1 = FT
              FT2 = (1.0-(1.0+RHO*SRATIO)/10.0)*FT*DEXP((-C*(EPSCR2-
     1            EPSCR1)))
              FT3 = FT*DEXP((-C*(EPSCR3-EPSCR1)))/2.0
              FT4 = FT/10.0
              AREA = 0.5*(FT1-FT2)*(EPSCR2-EPSCR1) + (EPSCR2-EPSCR1)*FT2
     1             + 0.5*(FT2-FT3)*(EPSCR3-EPSCR2) + (EPSCR3-EPSCR2)*FT3
     2             + 0.5*(FT3-FT4)*(EPSCR4-EPSCR3) + (EPSCR4-EPSCR3)*FT4
              IF (EPSN.LE.EPSCR2 .AND. EPSO.LE.EPSCR2) THEN
                  SUBARA = 0.5*(EPSN-EPSO)*FT1
              ELSE IF (EPSN.GT.EPSCR2 .AND. EPSN.LE.EPSCR3 .AND. EPSO
     1                .LE.EPSCR2) THEN
                  SIGN = EST*EPSN
                  SUBARA = 0.5*(EPSCR2-EPSO)*FT1 + 0.5*(EPSCR2-EPSCR1)*
     1                FT2 + 0.5*(EPSN-EPSCR2)*(FT2+SIGN) - 0.5*(EPSN-
```

```
     2             EPSCR1)*SIGN
            ELSE IF (EPSN.GT.EPSCR3 .AND. EPSN.LE.EPSCR4 .AND. EPSO
     1             .LE.EPSCR2) THEN
               SIGN = EST*EPSN
               SUBARA = 0.5*(EPSCR2-EPSO)*FT1 + 0.5*(EPSCR2-EPSCR1)*
     1             FT2 + 0.5*(EPSCR3-EPSCR2)*(FT2+FT3) + 0.5*(EPSN-
     2             EPSCR3)*(FT2+SIGN) - 0.5*(EPSN-EPSCR1)*SIGN
            ELSE IF (EPSN.GT.EPSCR2 .AND. EPSN.LE.EPSCR3 .AND. EPSO
     1             .GT.EPSCR2 .AND. EPSO.LE.EPSCR3) THEN
               SIGN = EST*EPSN
               SIGO = SPRING*EPSO
               SUBARA = 0.5*(EPSCR2-EPSCR1)*(FT1+FT2) + 0.5*(EPSN-
     1             EPSCR2)*(FT2+SIGN) - 0.5*(EPSN-EPSCR1)*SIGN - 0.5*
     2             (EPSCR2-EPSCR1)*(FT2+FT1) - 0.5*(EPSO-EPSCR2)*(
     3             SIGO+FT2) + 0.5*(EPSO-EPSCR1)*SIGO
            ELSE IF (EPSN.GT.EPSCR3 .AND. EPSN.LE.EPSCR4 .AND. EPSO
     1             .GT.EPSCR2 .AND. EPSO.LE.EPSCR3) THEN
               SIGN = EST*EPSN
               SIGO = SPRING*EPSO
               SUBARA = 0.5*(EPSCR2-EPSCR1)*(FT1+FT2) + 0.5*(EPSCR3-
     1             EPSCR2)*(FT3+FT2) + 0.5*(EPSN-EPSCR3)*(FT3+SIGN)
     2              - 0.5*(EPSN-EPSCR1)*SIGN - 0.5*(EPSCR2-EPSCR1)*(
     3             FT2+FT1) - 0.5*(EPSO-EPSCR2)*(SIGO+FT2) + 0.5*(
     4             EPSO-EPSCR1)*SIGO
            ELSE IF (EPSN.GT.EPSCR3 .AND. EPSN.LE.EPSCR4 .AND. EPSO
     1             .GT.EPSCR3 .AND. EPSO.LE.EPSCR4) THEN
               SIGN = EST*EPSN
               SIGO = SPRING*EPSO
               SUBARA = 0.5*(EPSCR2-EPSCR1)*(FT1+FT2) + 0.5*(EPSCR3-
     1             EPSCR2)*(FT2+FT3) + 0.5*(EPSN-EPSCR3)*(FT3+SIGN)
     2              - 0.5*(EPSN-EPSCR1)*SIGN - 0.5*(EPSCR2-EPSCR1)*(
     3             FT2+FT1) - 0.5*(EPSCR3-EPSCR2)*(FT3+FT2) - 0.5*(
     4             EPSO-EPSCR3)*(SIGO+FT3) + 0.5*(EPSO-EPSCR1)*SIGO
            ENDIF
         ENDIF
         DIFFER = (SUBARA/AREA)*100.0
C
         IF (DIFFER .LE. TOLER) RETURN
         IF (DIFFER .GT. TOLER) THEN
            TENSTF(IGAUSS,24+I,ILAYER) = EST
            IFLAG = 1
            CONVER = 999.0
            NSTIFF = 1
            CHKGUS = .TRUE.
         ENDIF
      ENDIF
      RETURN
  900       FORMAT(1X,'SUB.SPRINGCON1.OVERSOFTENING CRACK#',I3,
     1            'FOR GAUSSPOINT',I2,'LAYER#',I2,'OLDSTF',D12.5,
     2            'NEWSTF',D12.5,'DIFFERENCE=',D12.5)
      END
C
C************************************************************
C
C     INCLUDE(PROCESS)
      SUBROUTINE SPSTN(SPRING,EPSO,ILAYER,I,IGAUSS,EPSN,ELNUM,NNEL,IFG)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
```

```
C
C LYINFO    GTLTK     GETTHK
C
C*************************************************************
      INTEGER ELNUM
      REAL*8 LTHICK
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON/ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      DIMENSION LTHICK(9),ZS(9),DCS(3,3)
C
C         FIND THE STRAIN FOR A GIVEN TENSION STIFFNESS SPRING-1
C
      FT = TENSTF(IGAUSS,I+6,ILAYER)
      EPSY = TENSTF(IGAUSS,12+I,ILAYER)
      CKSTIN = TENSTF(IGAUSS,I+9,ILAYER)
C       IFG=2
      IF (IFG .EQ. 1) THEN
          FT1 = 0.35*FT
          FT2 = 0.10*FT
          EPSCR = CKSTIN
          EPSCR1 = EPSY*0.50
          EPSCR2 = EPSY
          STIF1 = FT1/EPSCR1
C
          IF (DABS(SPRING) .GE. DABS(STIF1)) THEN
              EPSO = (FT*EPSCR1-FT1*EPSCR)/(SPRING*(EPSCR1-EPSCR)+(FT-
     1            FT1))
              EPSO = DMAX1(CKSTIN,EPSO)
          ELSE
              EPSO = (FT1*EPSCR2-FT2*EPSCR1)/(SPRING*(EPSCR2-EPSCR1)+(
     1            FT1-FT2))
              EPSO = DMAX1(0.5*EPSY,EPSO)
          ENDIF
      ELSE IF (IFG .EQ. 2) THEN
          SIGYLD = TENSTF(IGAUSS,I+63,ILAYER)
          SRATIO = TENSTF(IGAUSS,I+60,ILAYER)
          ILL = TENSTF(IGAUSS,I+66,ILAYER)
          CALL LYINFO (ILL, LTHICK, ZS, MATRL, DCS, NNEL, ELNUM, NIPXI,
     1        NIPETA, NIPSI)
          CALL GTLTK (L, ELNUM, NNEL, THICKL, ZSI, LTHICK, ZS)
          CALL GETTHK (L, ELNUM, NNEL, THICKE, RAD)
          RHO = THICKL/THICKE
          EPSCR1 = SRATIO*FT*EPSY/SIGYLD
          EPSCR2 = (SRATIO*FT/SIGYLD)*(1.0+(1.0+RHO*SRATIO)/(10.0*RHO*
     1        SRATIO))*EPSY
          EPSCR3 = (1.0-(1.0/(2.*RHO*SRATIO))*(SRATIO*FT/SIGYLD))*EPSY
          EPSCR4 = EPSY
          C = 550.0
          FT1 = FT
          FT2=(1.0-(1.0+RHO*SRATIO)/10.0)*FT*DEXP((-C*(EPSCR2-EPSCR1)))
          FT3 = FT*DEXP((-C*(EPSCR3-EPSCR1)))/2.0
          FT4 = FT/10.0
          SPG1 = FT1/EPSCR1
          SPG2 = FT2/EPSCR2
          SPG3 = FT3/EPSCR3
          SPG4 = FT4/EPSCR4
          IF (SPRING .GE. SPG2) THEN
              EPSO = (FT2*EPSCR1-FT1*EPSCR2)/((FT2-FT1)+(EPSCR1-EPSCR2)*
     1            SPRING)
          ELSE IF (SPRING.LT.SPG2 .AND. SPRING.GE.SPG3) THEN
              EPSO = (FT3*EPSCR2-FT2*EPSCR3)/((FT3-FT2)+(EPSCR2-EPSCR3)*
     1            SPRING)
```

```
          ELSE IF (SPRING.LT.SPG3 .AND. SPRING.GE.SPG4) THEN
              EPSO = (FT4*EPSCR3-FT3*EPSCR4)/((FT4-FT3)+(EPSCR3-EPSCR4)*
     1            SPRING)
          ENDIF
      ENDIF
C           WRITE(*,*) 'SPGSTN STIFF',SPRING,'EPSOLD=',EPSO
      RETURN
      END
C
C***************************************************************
C
C     INCLUDE(PROCESS)
        SUBROUTINE CKTEN(EPSN,SPRING,IGAUSS,I,ILAYER,ELNUM,NNEL,IFG)
C  ***********************************************************
C
C     THIS SUBROUTINE CALCULATES THE STIFFNESS OF THE TENSION
C     STIFFNESS SPRING AT A GIVEN SAMPLEPOINT OF A GIVEN LAYER
C     OF A PARTICULAR ELEMENT.
C
C     INPUT PARAMETERS: FT=TENSILE STRESS IN CONCRETE AT SPRING
C                       INITIATION.
C                       EPSY=YEILD STRAIN OF ADJACENT STEEL LAYER
C                       EPST= CURRENT SPRING STRAIN.
C     OUTPUT RETURNED:  SPRING: SECANT STIFFNESS CORRSP. TO EPST.
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C LYINFO    GTLTK     GETTHK
C
C*************************************************************
      INTEGER ELNUM
      REAL*8 LTHICK
      COMMON/ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      DIMENSION LTHICK(9),ZS(9),DCS(3,3)
C
C   FIND THE TENSION STIFFNESS OF SPRING FOR GIVEN STRAIN-1
      FT = TENSTF(IGAUSS,I+6,ILAYER)
      EPSY = TENSTF(IGAUSS,12+I,ILAYER)
      CKSTIN = TENSTF(IGAUSS,I+9,ILAYER)
      IF (IFG .EQ. 1) THEN
          FT1 = 0.35*FT
          FT2 = 0.10*FT
          EPSCR = CKSTIN
          EPSCR1 = 0.50*EPSY
          EPSCR2 = EPSY
C
          IF (EPSN .LE. EPSCR1) THEN
              SPRING = (FT*(EPSCR1-EPSN)+FT1*(EPSN-EPSCR))/(EPSN*(EPSCR1
     1            -EPSCR))
          ELSE
              SPRING = (FT1*(EPSCR2-EPSN)+FT2*(EPSN-EPSCR1))/(EPSN*(
     1            EPSCR2-EPSCR1))
          ENDIF
      ELSE IF (IFG .EQ. 2) THEN
          SIGYLD = TENSTF(IGAUSS,I+63,ILAYER)
```

```
          SRATIO = TENSTF(IGAUSS,I+60,ILAYER)
          ILL = TENSTF(IGAUSS,I+66,ILAYER)
          CALL LYINFO (ILL, LTHICK, ZS, MATRL, DCS, NNEL, ELNUM, NIPXI,
     1        NIPETA, NIPSI)
          CALL GTLTK (L, ELNUM, NNEL, THICKL, ZSI, LTHICK, ZS)
          CALL GETTHK (L, ELNUM, NNEL, THICKE, RAD)
          RHO = THICKL/THICKE
          EPSCR1 = SRATIO*FT*EPSY/SIGYLD
          EPSCR2 = (SRATIO*FT/SIGYLD)*(1.0+(1.0+RHO*SRATIO)/(10.0*RHO*
     1        SRATIO))*EPSY
          EPSCR3 = (1.0-(1.0/(2.*RHO*SRATIO))*(SRATIO*FT/SIGYLD))*EPSY
          EPSCR4 = EPSY
          C = 550.0
          FT1 = FT
          FT2=(1.0-(1.0+RHO*SRATIO)/10.0)*FT*DEXP((-C*(EPSCR2-EPSCR1)))
          FT3 = FT*DEXP((-C*(EPSCR3-EPSCR1)))/2.0
          FT4 = FT/10.0
C
          IF (EPSN .LE. EPSCR2) THEN
              SPRING = ((FT1-FT2)*EPSN+(FT2*EPSCR1-FT1*EPSCR2))/((EPSCR1
     1            -EPSCR2)*EPSN)
          ELSE IF (EPSN.GT.EPSCR2 .AND. EPSN.LE.EPSCR3) THEN
              SPRING = ((FT2-FT3)*EPSN+(FT3*EPSCR2-FT2*EPSCR3))/((EPSCR2
     1            -EPSCR3)*EPSN)
          ELSE IF (EPSN.GT.EPSCR3 .AND. EPSN.LE.EPSCR4) THEN
              SPRING = ((FT3-FT4)*EPSN+(FT4*EPSCR3-FT3*EPSCR4))/((EPSCR3
     1            -EPSCR4)*EPSN)
          ENDIF
      ENDIF
      RETURN
      END
C
C************************************************************
C
C     INCLUDE(PROCESS)
        SUBROUTINE COLLIN(EPSMAX,EPSMIN,PROPER,IGAUSS,ILAYER,SIGMOD
     $                   ,STIFM,FC2MAX,IOUT)
        IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************


C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C************************************************************
C
        DIMENSION PROPER(25)
C
      EINITL = PROPER(1)
      FC = PROPER(5)
      EPSC = PROPER(7)
C
      FC2MAX = 1.0/(0.8+0.34*(EPSMAX/EPSC))
      IF (FC2MAX .GE. 1.0) FC2MAX = 1.0
C
      FC2MAX = FC*FC2MAX
      SIGMOD = FC2MAX*(2.0*(DABS(EPSMIN)/EPSC)-(DABS(EPSMIN)/EPSC)**2)
      STIFM = SIGMOD/DABS(EPSMIN)
      STIFM = DMAX1(EINITL/1000.0,STIFM)
C
```

```
      RETURN
      END
C
C************************************************************
C
C   COMPRESSION SOFTENING AFTER CRACKING
C
C     INCLUDE(PROCESS)
      SUBROUTINE CMPSFT(EPSMAX,EPSMID,EPSMIN,SIGMA1,SIGMA2,SIGMA3,
     $       PROPER,EPS1,EPS2,EPS3,L,J,IOUT,CONVER,NSTIFF,CHKGUS,TOLER)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C VECHIO
C
C***********************************************************
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON/ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      DIMENSION PROPER(25)
C     REAL*8 NUS
      LOGICAL CHKGUS
      EMAXOD = HISTO(L,38,J)
      EMIDOD = HISTO(L,39,J)
      EMINOD = HISTO(L,40,J)
C           WRITE(*,*) 'MAT STATE IN COMP.SOFT', HISTO(L,29,J)
      IF (HISTO(L,29,J) .EQ. 4.0) RETURN
      IF (DABS(EPSMAX).GE.0.999*DABS(EMAXOD) .OR. DABS(EPSMID).GE.0.999*
     1    DABS(EMIDOD) .OR. DABS(EPSMIN).GE.0.999*DABS(EMINOD)) THEN
          HISTO(L,30,J) = 1.0
          HISTO(L,38,J) = EPSMAX
          HISTO(L,39,J) = EPSMID
          HISTO(L,40,J) = EPSMIN
          HISTO(L,41,J) = EPSMAX
          HISTO(L,42,J) = EPSMID
          HISTO(L,43,J) = EPSMIN
          HISTO(L,22,J) = EPS1
          HISTO(L,23,J) = EPS2
          HISTO(L,24,J) = EPS3
          HISTO(L,31,J) = EPS1
          HISTO(L,32,J) = EPS2
          HISTO(L,33,J) = EPS3
          CALL VECHIO (L, J, EPSMAX, EPSMID, EPSMIN, PROPER, EPS1, EPS2
     1        , EPS3, SIGMA1, SIGMA2, SIGMA3, CONVER, NSTIFF, CHKGUS,
     2        TOLER, IOUT, NOPRNT)
          RETURN
      ELSE
          IFLAG3 = 1
          PEPSMX = HISTO(L,41,J)
          PEPSMD = HISTO(L,42,J)
          PEPSMN = HISTO(L,43,J)
          IF (DABS(EPSMAX).LT.DABS(PEPSMX) .AND. DABS(EPSMID).LT.DABS(
     1        PEPSMD) .AND. DABS(EPSMIN).LT.DABS(PEPSMN)) THEN
              HISTO(L,30,J) = 2.0
              VARY1 = EPSMAX*PEPSMX
              VARY2 = EPSMID*PEPSMX
              VARY3 = EPSMIN*PEPSMN
              IF (VARY1.LT.0.0 .OR. VARY2.LT.0.0 .OR. VARY3.LT.0.0) THEN
                  IFLAG3 = 0
```

```
                CONVER = 999.0
                NSTIFF = 1
                CHKGUS = .TRUE.
                IF (VARY1 .LT. 0.0) THEN
                    HISTO(L,38,J) = 0.0
                    HISTO(L,41,J) = 0.0
                ELSE IF (VARY2 .LT. 0.0) THEN
                    HISTO(L,39,J) = 0.0
                    HISTO(L,42,J) = 0.0
                ELSE IF (VARY3 .LT. 0.0) THEN
                    HISTO(L,40,J) = 0.0
                    HISTO(L,43,J) = 0.0
                ENDIF
            ENDIF
        ELSE
            HISTO(L,30,J) = 3.0
        ENDIF
        IF (IFLAG3 .EQ. 1) THEN
            HISTO(L,41,J) = EPSMAX
            HISTO(L,42,J) = EPSMID
            HISTO(L,43,J) = EPSMIN
            HISTO(L,31,J) = EPS1
            HISTO(L,32,J) = EPS2
            HISTO(L,33,J) = EPS3
        ENDIF
      ENDIF
      RETURN
      END
C*******************************************************************
C     INCLUDE(PROCESS)
        SUBROUTINE VECHIO(IGAUSS,ILAYER,EPSMAX,EPSMID,EPSMIN,PROPER,
     $       EPS1,EPS2,EPS3,SIGMA1,SIGMA2,SIGMA3,CONVER,NSTIFF,CHKGUS,
     $       TOLER,IOUT)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C COLLIN
C
C*************************************************************
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON/ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
        DIMENSION PROPER(25)
        LOGICAL CHKGUS
C
      TOLERE = TOLER
      EINITL = PROPER(1)
      IF (HISTO(IGAUSS,30,ILAYER) .EQ. 1.0) THEN
          IF (HISTO(IGAUSS,29,ILAYER) .EQ. 1.0) THEN
              EPSC = PROPER(7)
              IF (DABS(EPSMIN) .LE. EPSC) THEN
C
                  STATE = 9999
                  HISTO(IGAUSS,25,ILAYER) = STATE
C
C       CHECK FAILURE BY CONCRETE REACHING ULTIMATE(COMPRESSIVE).
C
                  IFLAG = 0
                  FMAX = HISTO(IGAUSS,58,ILAYER)
```

```
C
              CALL COLLIN (EPSMAX, EPSMIN, PROPER, IGAUSS, ILAYER,
     1           SIGMOD, STIFM, FC2MAX, IOUT)
C
              C1=100*(HISTO(IGAUSS,20,ILAYER)**2-STIFM**2)/STIFM**2
              IF (C1 .GE. TOLERE) THEN
                 HISTO(IGAUSS,20,ILAYER) = (1.0-TOLERE/200.)*HISTO(
     1              IGAUSS,20,ILAYER)
                 HISTO(IGAUSS,58,ILAYER) = FC2MAX
                 NSTIFF = 1
                 CONVER = 999.0
                 WRITE (*, *) 'VEC 1 999'
                 CHKGUS = .TRUE.
                 IFLAG = 1
              ENDIF
              IF (IFLAG .EQ. 0) THEN
                 STRESS = HISTO(IGAUSS,20,ILAYER)*EPSMIN
                 IF (DABS(STRESS) .GE. DABS(FMAX)) THEN
                    EPSC = PROPER(7)
                    SIGMOD = FMAX*(2.0*(DABS(EPSMIN)/EPSC)-(DABS(
     1                 EPSMIN)/EPSC)**2)
                    ENEW = DABS(SIGMOD)/DABS(EPSMIN)
                    IF(ENEW.LE.EINITL/1000.0)ENEW=EINITL/1000.
                    HISTO(IGAUSS,20,ILAYER) = ENEW
                    CHKGUS = .TRUE.
                    NSTIFF = 1
                    CONVER = 999
                    WRITE (*, *) 'VEC 2 999'
C                                  WRITE(*,20)IFLAG,C1,TOLER,
C    $                             HISTO(IGAUSS,20,ILAYER)
                 ENDIF
              ENDIF
           ENDIF
           IF (DABS(EPSMIN) .GT. DABS(EPSC)) THEN
              HISTO(IGAUSS,29,ILAYER) = 2.0
              ENEW = HISTO(IGAUSS,58,ILAYER)/DABS(EPSC)
              IF (ENEW .LE. EINITL/1000.) ENEW = EINITL/1000.0
              HISTO(IGAUSS,20,ILAYER) = ENEW
              CONVER = 999.0
              WRITE (*, *) 'VEC 3 999'
              CHKGUS = .TRUE.
              NSTIFF = 1
           ENDIF
           RETURN
        ENDIF
        IF (HISTO(IGAUSS,29,ILAYER) .EQ. 2.0) THEN
           IF (DABS(SIGMA3) .GE. DABS(HISTO(IGAUSS,58,ILAYER))) THEN
              HISTO(IGAUSS,29,ILAYER) = 3.0
              TOLERE = 1.0
              HISTO(IGAUSS,20,ILAYER) = (1.0-TOLERE/200.)*HISTO(
     1           IGAUSS,20,ILAYER)
              HISTO(IGAUSS,20,ILAYER) = DMAX1(EINITL/1000.0,HISTO(
     1           IGAUSS,20,ILAYER))
              HISTO(IGAUSS,57,ILAYER) = HISTO(IGAUSS,20,ILAYER)
              HISTO(IGAUSS,59,ILAYER) = HISTO(IGAUSS,21,ILAYER)
              HISTO(IGAUSS,34,ILAYER) = SIGMA1
              HISTO(IGAUSS,35,ILAYER) = SIGMA2
              HISTO(IGAUSS,36,ILAYER) = HISTO(IGAUSS,58,ILAYER)
              CONVER = 999.0
              WRITE (*, *) 'VEC 4 999'
              NSTIFF = 1
              CHKGUS = .TRUE.
           ELSE
```

```
              HISTO(IGAUSS,29,ILAYER) = 1.0
              HISTO(IGAUSS,57,ILAYER) = HISTO(IGAUSS,20,ILAYER)
              HISTO(IGAUSS,59,ILAYER) = HISTO(IGAUSS,21,ILAYER)
              HISTO(IGAUSS,34,ILAYER) = SIGMA1
              HISTO(IGAUSS,35,ILAYER) = SIGMA2
              HISTO(IGAUSS,36,ILAYER) = HISTO(IGAUSS,58,ILAYER)
              CONVER = 999.0
              WRITE (*, *) 'VEC 5 999'
          ENDIF
          RETURN
      ENDIF
      IF (HISTO(IGAUSS,29,ILAYER) .EQ. 3.0) THEN
          IF (DABS(EPSMIN) .LT. PROPER(8)) THEN
              IFLAG = 0
              FMAX = HISTO(IGAUSS,58,ILAYER)
              CALL COLLIN (EPSMAX, EPSMIN, PROPER, IGAUSS, ILAYER,
     1            SIGMOD, STIFM, FC2MAX, IOUT)
              STIFM = DMAX1(EINITL/1000.0,STIFM)
              C1 = 100.0*(HISTO(IGAUSS,20,ILAYER)**2-STIFM**2)/STIFM
     1            **2
              IF (C1 .GE. TOLERE) THEN
                  HISTO(IGAUSS,20,ILAYER) = HISTO(IGAUSS,20,ILAYER)*
     1                (1.0-TOLERE/200)
                  HISTO(IGAUSS,58,ILAYER) = FC2MAX
                  HISTO(IGAUSS,57,ILAYER) = HISTO(IGAUSS,20,ILAYER)
                  HISTO(IGAUSS,59,ILAYER) = HISTO(IGAUSS,21,ILAYER)
                  CONVER = 999.0
                  WRITE (*, *) 'VEC 6 999'
                  NSTIFF = 1
                  CHKGUS = .TRUE.
                  IFLAG = 1
              ENDIF
              IF (IFLAG .EQ. 0) THEN
                  STRESS = HISTO(IGAUSS,20,ILAYER)*EPSMIN
                  IF (DABS(STRESS) .GE. DABS(FMAX)) THEN
                      EPSC = PROPER(7)
                      SIGMOD = FMAX*(2.0*(DABS(EPSMIN)/EPSC)-(DABS(
     1                    EPSMIN)/EPSC)**2)
                      ENEW = DABS(SIGMOD)/DABS(EPSMIN)
                      ENEW = DMAX1(EINITL/1000.0,ENEW)
                      HISTO(IGAUSS,20,ILAYER) = ENEW
                      HISTO(IGAUSS,57,ILAYER) = HISTO(IGAUSS,20,
     1                    ILAYER)
                      HISTO(IGAUSS,59,ILAYER) = HISTO(IGAUSS,21,
     1                    ILAYER)
                      CHKGUS = .TRUE.
                      NSTIFF = 1
                      CONVER = 999
                      WRITE (*, *) 'VEC 9 999'
                  ENDIF
              ENDIF
              RETURN
          ENDIF
      ENDIF
      IF (DABS(EPSMIN) .GE. PROPER(8)) THEN
          HISTO(IGAUSS,29,ILAYER) = 4.0
          CRACK(IGAUSS,1,ILAYER) = 0.0
          HISTO(IGAUSS,20,ILAYER) = PROPER(1)/1000.0
          HISTO(IGAUSS,21,ILAYER) = 0.001
          CONVER = 999.0
          WRITE (*, *) 'VEC10 999'
          CHKGUS = .TRUE.
          NSTIFF = 1
```

```
            RETURN
         ENDIF
      ENDIF
C
  100 CONTINUE
      RETURN
      END
C*************************************************************
C     INCLUDE(PROCESS)
      SUBROUTINE VINTAS(GMCRK1,GMCRK2,EPSCRK,SIGMA1,SIGMA2,SIGMA3,PDIR,
     $                  ICRACK,L,J,IOUT,ELNUM,NNEL,NLAYRS)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C SHSTIF   LCHECK    LYINFO    GETTHK    GTLTK     DTRANS
C
C************************************************************
      INTEGER ELNUM
      REAL*8  LTHICK
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON/ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      DIMENSION DCON(6,6),DCS(3,3),LTHICK(9),ZS(9)
      REAL*8 NUS,NT(3,3),N(3,3),BSTL(3,3),GCRK(6,6),PDIR(3,3)
C
C THIS SUBROUTINE IS UTILIZED TO COMPUTE THE SHEAR STIFFNESS OF THE
C CRACK BASED ON EXPERIMENTS BY VINTZELEOU ET. AL. AND CORRECTING IT
C WITH THE EFFECTS OF REINFORCEMENT AND THE LOCALIZED BOND PRESENT
C AT THE POINT.
C
      ICG = 9*(ICRACK-1) + 1
      N(1,1) = CRACK(L,137+ICG,J)
      N(1,2) = CRACK(L,138+ICG,J)
      N(1,3) = CRACK(L,139+ICG,J)
      N(2,1) = CRACK(L,140+ICG,J)
      N(2,2) = CRACK(L,141+ICG,J)
      N(2,3) = CRACK(L,142+ICG,J)
      N(3,1) = CRACK(L,143+ICG,J)
      N(3,2) = CRACK(L,144+ICG,J)
      N(3,3) = CRACK(L,145+ICG,J)
C
      ICOUNT = 0
      SIGBND = 0.0
C
      DO 120 IN = 1, 3
C
         IF (TENSTF(L,IN,J) .EQ. 1.0) THEN
            ISG = TENSTF(L,3+IN,J)
            STRES = TENSTF(L,18+ISG,J)*TENSTF(L,24+ISG,J)
            IMMG = (ISG-1)*9 + 1
            NT(1,1) = TENSTF(L,30+IMMG,J)
            NT(2,1) = TENSTF(L,31+IMMG,J)
            NT(3,1) = TENSTF(L,32+IMMG,J)
            NT(1,2) = TENSTF(L,33+IMMG,J)
            NT(2,2) = TENSTF(L,34+IMMG,J)
            NT(3,2) = TENSTF(L,35+IMMG,J)
            NT(1,3) = TENSTF(L,36+IMMG,J)
            NT(2,3) = TENSTF(L,37+IMMG,J)
            NT(3,3) = TENSTF(L,38+IMMG,J)
```

```
              DO 110 IM = 1, 3
                 DO 100 IP = 1, 3
                    BSTL(IM,IP) = 0.0
                    BSTL(IM,IP) = BSTL(IM,IP) + N(IM,1)*NT(1,IP)
                    BSTL(IM,IP) = BSTL(IM,IP) + N(IM,2)*NT(2,IP)
                    BSTL(IM,IP) = BSTL(IM,IP) + N(IM,3)*NT(3,IP)
  100            CONTINUE
  110         CONTINUE
              STRESN = BSTL(1,1)*BSTL(1,1)*STRES
              SIGBND = SIGBND + STRESN
           ELSE
              ICOUNT = ICOUNT + 1
           ENDIF
  120 CONTINUE
      IF (ICOUNT .EQ. 3) SIGBND = CRACK(L,37+ICRACK,J)
C
      GCRAK1 = 0.0
      GCRAK2 = 0.0
      IF (DABS(GMCRK1) .LE. 1.E-6) GMCRK1 = 1.0E-6
      CALL SHSTIF (GMCRK1, SIGBND, GCRAK1, IOUT, ELNUM, L, J)
  130 CONTINUE
      IF (DABS(GMCRK2) .LE. 1.E-6) GMCRK2 = 1.0E-6
      CALL SHSTIF (GMCRK2, SIGBND, GCRAK2, IOUT, ELNUM, L, J)
  140 CONTINUE
      IF (DABS(GMCRK1).LE.1.E-9 .AND. DABS(GMCRK2).LE.1.E-9) RETURN
C
      DO 160 KJM = 1, 6
         DO 150 KPM = 1, 6
            GCRK(KJM,KPM) = 0.0
  150    CONTINUE
  160 CONTINUE
C
C       THE PERCENTAGE RATIO OF THE CURRENT LAYER IS
C       NOW CALCULATED AT CURRENT INTEGRATION POINT BASED ON LAYER TO
C       TOTAL THICKNESS.
C
      DO 170 JJK = 1, NLAYRS
         ROFAC = 0.0
         CALL LCHECK (ELNUM, JJK, IFCHK)
         IF (IFCHK .EQ. 0) THEN
            CALL LYINFO (JJK, LTHICK, ZS, MATRL, DCS, NNEL, ELNUM,
     1           NIPXI, NIPETA, NIPSI)
            CALL GETTHK (L, ELNUM, NNEL, THICKE, RAD)
            CALL GTLTK (L, ELNUM, NNEL, THICKL, ZSI, LTHICK, ZS)
            IF (JJK .EQ. J+1) THEN
               ROFAC = THICKL/THICKE
            ELSE IF (JJK .EQ. J+2) THEN
               CALL LCHECK (ELNUM, J + 1, IFCHK)
               IF (IFCHK .EQ. 0) ROFAC = THICKL/THICKE
            ELSE IF (JJK .EQ. J-1) THEN
               ROFAC = THICKL/THICKE
            ELSE IF (JJK .EQ. J-2) THEN
               CALL LCHECK (ELNUM, J - 1, IFCHK)
               IF (IFCHK .EQ. 0) ROFAC = THICKL/THICKE
            ELSE
               ROFAC = 0.0
            ENDIF
         ELSE IF (JJK .EQ. J) THEN
            ROFAC = 1.0
         ELSE
            ROFAC = 0.0
         ENDIF

         GCRK(1,1) = ROFAC*CRACK(L,2,JJK) + GCRK(1,1)
```

```
        GCRK(1,2) = ROFAC*CRACK(L,3,JJK) + GCRK(1,2)
        GCRK(1,3) = ROFAC*CRACK(L,4,JJK) + GCRK(1,3)
        GCRK(1,4) = ROFAC*CRACK(L,5,JJK) + GCRK(1,4)
        GCRK(1,5) = ROFAC*CRACK(L,6,JJK) + GCRK(1,5)
        GCRK(1,6) = ROFAC*CRACK(L,7,JJK) + GCRK(1,6)
        GCRK(2,1) = ROFAC*CRACK(L,8,JJK) + GCRK(2,1)
        GCRK(2,2) = ROFAC*CRACK(L,9,JJK) + GCRK(2,2)
        GCRK(2,3) = ROFAC*CRACK(L,10,JJK) + GCRK(2,3)
        GCRK(2,4) = ROFAC*CRACK(L,11,JJK) + GCRK(2,4)
        GCRK(2,5) = ROFAC*CRACK(L,12,JJK) + GCRK(2,5)
        GCRK(2,6) = ROFAC*CRACK(L,13,JJK) + GCRK(2,6)
        GCRK(3,1) = ROFAC*CRACK(L,14,JJK) + GCRK(3,1)
        GCRK(3,2) = ROFAC*CRACK(L,15,JJK) + GCRK(3,2)
        GCRK(3,3) = ROFAC*CRACK(L,16,JJK) + GCRK(3,3)
        GCRK(3,4) = ROFAC*CRACK(L,17,JJK) + GCRK(3,4)
        GCRK(3,5) = ROFAC*CRACK(L,18,JJK) + GCRK(3,5)
        GCRK(3,6) = ROFAC*CRACK(L,19,JJK) + GCRK(3,6)
        GCRK(4,1) = ROFAC*CRACK(L,20,JJK) + GCRK(4,1)
        GCRK(4,2) = ROFAC*CRACK(L,21,JJK) + GCRK(4,2)
        GCRK(4,3) = ROFAC*CRACK(L,22,JJK) + GCRK(4,3)
        GCRK(4,4) = ROFAC*CRACK(L,23,JJK) + GCRK(4,4)
        GCRK(4,5) = ROFAC*CRACK(L,24,JJK) + GCRK(4,5)
        GCRK(4,6) = ROFAC*CRACK(L,25,JJK) + GCRK(4,6)
        GCRK(5,1) = ROFAC*CRACK(L,26,JJK) + GCRK(5,1)
        GCRK(5,2) = ROFAC*CRACK(L,27,JJK) + GCRK(5,2)
        GCRK(5,3) = ROFAC*CRACK(L,28,JJK) + GCRK(5,3)
        GCRK(5,4) = ROFAC*CRACK(L,29,JJK) + GCRK(5,4)
        GCRK(5,5) = ROFAC*CRACK(L,30,JJK) + GCRK(5,5)
        GCRK(5,6) = ROFAC*CRACK(L,31,JJK) + GCRK(5,6)
        GCRK(6,1) = ROFAC*CRACK(L,32,JJK) + GCRK(6,1)
        GCRK(6,2) = ROFAC*CRACK(L,33,JJK) + GCRK(6,2)
        GCRK(6,3) = ROFAC*CRACK(L,34,JJK) + GCRK(6,3)
        GCRK(6,4) = ROFAC*CRACK(L,35,JJK) + GCRK(6,4)
        GCRK(6,5) = ROFAC*CRACK(L,36,JJK) + GCRK(6,5)
        GCRK(6,6) = ROFAC*CRACK(L,37,JJK) + GCRK(6,6)
170 CONTINUE
    CALL DTRANS (GCRK, N, IOUT)
    ECURNT = GCRK(1,1)
    DO 190 KJM = 1, 6
        DO 180 KPM = 1, 6
            DCON(KJM,KPM) = 0.0
            GCRK(KJM,KPM) = 0.0
180     CONTINUE
190 CONTINUE
    DO 200 JJK = 1, NLAYRS
        ROFAC = 0.0
        CALL LCHECK (ELNUM, JJK, IFCHK)
        IF (IFCHK .EQ. 0) THEN
            CALL LYINFO (JJK, LTHICK, ZS, MATRL, DCS, NNEL, ELNUM,
   1            NIPXI, NIPETA, NIPSI)
            CALL GETTHK (L, ELNUM, NNEL, THICKE, RAD)
            CALL GTLTK (L, ELNUM, NNEL, THICKL, ZSI, LTHICK, ZS)
            IF (JJK .EQ. J+1) THEN
                ROFAC = THICKL/THICKE
            ELSE IF (JJK .EQ. J+2) THEN
                CALL LCHECK (ELNUM, J + 1, IFCHK)
                IF (IFCHK .EQ. 0) ROFAC = THICKL/THICKE
            ELSE IF (JJK .EQ. J-1) THEN
                ROFAC = THICKL/THICKE
            ELSE IF (JJK .EQ. J-2) THEN
                CALL LCHECK (ELNUM, J - 1, IFCHK)
                IF (IFCHK .EQ. 0) ROFAC = THICKL/THICKE
            ELSE
```

```
                ROFAC = 0.0
             ENDIF
          ELSE
             ROFAC = 0.0
          ENDIF
          GCRK(1,1) = ROFAC*CRACK(L,2,JJK) + GCRK(1,1)
          GCRK(1,2) = ROFAC*CRACK(L,3,JJK) + GCRK(1,2)
          GCRK(1,3) = ROFAC*CRACK(L,4,JJK) + GCRK(1,3)
          GCRK(1,4) = ROFAC*CRACK(L,5,JJK) + GCRK(1,4)
          GCRK(1,5) = ROFAC*CRACK(L,6,JJK) + GCRK(1,5)
          GCRK(1,6) = ROFAC*CRACK(L,7,JJK) + GCRK(1,6)
          GCRK(2,1) = ROFAC*CRACK(L,8,JJK) + GCRK(2,1)
          GCRK(2,2) = ROFAC*CRACK(L,9,JJK) + GCRK(2,2)
          GCRK(2,3) = ROFAC*CRACK(L,10,JJK) + GCRK(2,3)
          GCRK(2,4) = ROFAC*CRACK(L,11,JJK) + GCRK(2,4)
          GCRK(2,5) = ROFAC*CRACK(L,12,JJK) + GCRK(2,5)
          GCRK(2,6) = ROFAC*CRACK(L,13,JJK) + GCRK(2,6)
          GCRK(3,1) = ROFAC*CRACK(L,14,JJK) + GCRK(3,1)
          GCRK(3,2) = ROFAC*CRACK(L,15,JJK) + GCRK(3,2)
          GCRK(3,3) = ROFAC*CRACK(L,16,JJK) + GCRK(3,3)
          GCRK(3,4) = ROFAC*CRACK(L,17,JJK) + GCRK(3,4)
          GCRK(3,5) = ROFAC*CRACK(L,18,JJK) + GCRK(3,5)
          GCRK(3,6) = ROFAC*CRACK(L,19,JJK) + GCRK(3,6)
          GCRK(4,1) = ROFAC*CRACK(L,20,JJK) + GCRK(4,1)
          GCRK(4,2) = ROFAC*CRACK(L,21,JJK) + GCRK(4,2)
          GCRK(4,3) = ROFAC*CRACK(L,22,JJK) + GCRK(4,3)
          GCRK(4,4) = ROFAC*CRACK(L,23,JJK) + GCRK(4,4)
          GCRK(4,5) = ROFAC*CRACK(L,24,JJK) + GCRK(4,5)
          GCRK(4,6) = ROFAC*CRACK(L,25,JJK) + GCRK(4,6)
          GCRK(5,1) = ROFAC*CRACK(L,26,JJK) + GCRK(5,1)
          GCRK(5,2) = ROFAC*CRACK(L,27,JJK) + GCRK(5,2)
          GCRK(5,3) = ROFAC*CRACK(L,28,JJK) + GCRK(5,3)
          GCRK(5,4) = ROFAC*CRACK(L,29,JJK) + GCRK(5,4)
          GCRK(5,5) = ROFAC*CRACK(L,30,JJK) + GCRK(5,5)
          GCRK(5,6) = ROFAC*CRACK(L,31,JJK) + GCRK(5,6)
          GCRK(6,1) = ROFAC*CRACK(L,32,JJK) + GCRK(6,1)
          GCRK(6,2) = ROFAC*CRACK(L,33,JJK) + GCRK(6,2)
          GCRK(6,3) = ROFAC*CRACK(L,34,JJK) + GCRK(6,3)
          GCRK(6,4) = ROFAC*CRACK(L,35,JJK) + GCRK(6,4)
          GCRK(6,5) = ROFAC*CRACK(L,36,JJK) + GCRK(6,5)
          GCRK(6,6) = ROFAC*CRACK(L,37,JJK) + GCRK(6,6)
200 CONTINUE
    ES = HISTO(L,20,J)
    NUS = HISTO(L,21,J)
    CONST = ES/((1.-2.0*NUS)*(1+NUS))
    DCON(1,1) = CONST*(1.0-NUS)
    DCON(2,2) = CONST*(1.0-NUS)
    DCON(3,3) = CONST*(1.0-NUS)
    DCON(1,2) = CONST*NUS
    DCON(2,1) = DCON(1,2)
    DCON(3,1) = DCON(1,2)
    DCON(1,3) = DCON(1,2)
    DCON(2,3) = DCON(1,2)
    DCON(3,2) = DCON(1,2)
    DCON(4,4) = ES/(2.0*(1+NUS))
    DCON(5,5) = DCON(4,4)
    DCON(6,6) = DCON(4,4)
    DO 220 KKP = 1, 6
        DO 210 KKJ = 1, 6
            GCRK(KKP,KKJ) = GCRK(KKP,KKJ) + DCON(KKP,KKJ)
210     CONTINUE
220 CONTINUE
    CALL DTRANS (GCRK, N, IOUT)
```

```
      IF (DABS(GMCRK1) .GT. 1.E-7) GCRAK1 = GCRAK1*(ECURNT/GCRK(1,1))
      IF (DABS(GMCRK2) .GT. 1.E-7) GCRAK2 = GCRAK2*(ECURNT/GCRK(1,1))
      GFLR = HISTO(L,26,J)/(2.0*(1.0+HISTO(L,27,J)))
      IF (GCRAK1 .LE. 0.01*GFLR) GCRAK1 = 0.010*GFLR
      IF (GCRAK2 .LE. 0.01*GFLR) GCRAK2 = 0.010*GFLR
      GCRAK1 = DMIN1(10.0*GFLR,GCRAK1)
      GCRAK2 = DMIN1(10.0*GFLR,GCRAK2)
      GCRAK1 = DMIN1(CRACK(L,97+ICRACK,J),GCRAK1)
      GCRAK2 = DMIN1(CRACK(L,107+ICRACK,J),GCRAK2)
      CRACK(L,97+ICRACK,J) = GCRAK1
      CRACK(L,107+ICRACK,J) = GCRAK2
      RETURN
      END
C**********************************************************
C
C       INCLUDE(PROCESS)
        SUBROUTINE LCHECK(ELNUM,ILAYER,IFCHK)
        IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C*************************************************************
        INTEGER ELNUM
C
        COMMON/LPROP/PROPER(25,15)
        COMMON/LAYERA/ELLYFO(320,5000)
C
C SUBROUTINE RETURNS A FLAG TO IDENTIFY GIVEN LAYER AS STEEL/OTHERS.
C
      KK = 21*(ILAYER-1) + 1
      MATRL = ELLYFO(KK,ELNUM)
      IFCHK = PROPER(25,MATRL)
C
      RETURN
      END
C************************************************************
C     INCLUDE(PROCESS)
        SUBROUTINE SHSTIF(GAMCRK,SIGBND,GCRACK,IOUT,ELNUM,L,J)
        IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C*************************************************************
        INTEGER ELNUM
        DIMENSION A(20),B(20),C(20),D(20),E(20)
C
C      UPDATE THE SHEAR STIFFNESS ACCROSS A CRACK BASED ON THE
C      EXPERIMENTS DONE BY VINTZELEOU AND TASSIOS ASCE/ACI 1987.
C
C       A -- SHEAR STRAIN DATA
C       B -- SHEAR STRESS WITH SIGNORMAL=0.0 (DOWEL TESTS)
C       C -- SHEAR STRESS WITH SIGNORMAL=0.5MPA (AGG. INTERLOCK)
C       D -- SHEAR STRESS WITH SIGNORMAL=1.0MPA (AGG. INTERLOCK)
```

```
C        E -- SHEAR STRESS WITH SIGNORMAL=2.0MPA (AGG. INTERLOCK)
C
      A(1) = 0.0
      A(2) = 4.16E-4
      A(3) = 8.33E-4
      A(4) = 1.25E-3
      A(5) = 1.66E-3
      A(6) = 2.50E-3
      A(7) = 3.33E-3
      A(8) = 4.16E-3
      A(9) = 5.00E-3
      A(10) = 5.83E-3
      A(11) = 6.66E-3
      A(12) = 7.50E-3
      A(13) = 8.33E-3
      A(14) = 9.16E-3
      A(15) = 9.99E-3
      A(16) = 0.01083
      A(17) = 0.01166
      A(18) = 0.01250
      A(19) = 0.01333
      B(1) = 0.0
      B(2) = 0.644
      B(3) = 0.777
      B(4) = 0.833
      B(5) = 0.888
      B(6) = 0.978
      B(7) = 1.056
      B(8) = 1.133
      B(9) = 1.20
      B(10) = 1.255
      B(11) = 1.288
      B(12) = 1.34
      B(13) = 1.388
      B(14) = 1.422
      B(15) = 1.433
      B(16) = 1.44
      B(17) = 1.466
      B(18) = 1.488
      B(19) = 1.50
      C(1) = 0.0
      C(2) = 1.50
      C(3) = 1.60
      C(4) = 1.67
      C(5) = 1.75
      C(6) = 1.85
      C(7) = 2.0
      C(8) = 2.10
      C(9) = 2.25
      C(10) = 2.35
      C(11) = 2.40
      C(12) = 2.45
      C(13) = 2.50
      C(14) = 2.50
      C(15) = 2.50
      C(16) = 2.50
      C(17) = 2.50
      C(18) = 2.50
      C(19) = 2.50
      D(1) = 0.0
      D(2) = 2.00
      D(3) = 2.50
      D(4) = 2.75
```

```
      D(5) = 2.80
      D(6) = 3.0
      D(7) = 3.15
      D(8) = 3.30
      D(9) = 3.50
      D(10) = 3.55
      D(11) = 3.60
      D(12) = 3.70
      D(13) = 3.75
      D(14) = 3.80
      D(15) = 3.85
      D(16) = 3.87
      D(17) = 3.90
      D(18) = 3.95
      D(19) = 4.00
      E(1) = 0.0
      E(2) = 2.00
      E(3) = 2.85
      E(4) = 3.25
      E(5) = 3.90
      E(6) = 4.60
      E(7) = 5.0
      E(8) = 5.30
      E(9) = 5.60
      E(10) = 5.85
      E(11) = 5.80
      E(12) = 5.75
      E(13) = 5.70
      E(14) = 5.60
      E(15) = 5.50
      E(16) = 5.40
      E(17) = 5.25
      E(18) = 5.10
      E(19) = 5.00
      TAUCRK = 0.0
      GCRACK = 0.0
      GAM1 = 0.0
      GAM2 = 0.0
      TAU1 = 0.0
      TAU2 = 0.0
      TAU3 = 0.0
      TAU4 = 0.0
      TAU5 = 0.0
      TAU6 = 0:0
      TAU7 = 0.0
      TAU8 = 0.0
      TAU12C = 0.0
      TAU34C = 0.0
      TAU56C = 0.0
      TAU78C = 0.0
      FX0 = 0.0
      FX01 = 0.0
      FX12 = 0.0
      FX34 = 0.0
      FX012 = 0.0
      FX123 = 0.0
      FX0123 = 0.0
C
C       FOR CURRENT STRAIN FIND THE UPPER AND LOWER EXPT. VALUES
C       OF THE STRAIN AND THE CORRSP. STRESSES, FOR EACH NORMAL
C       LOAD CURVE GIVEN HERE. FIND STRESS VALUES FOR CURRENT STRAIN
C       ON EACH OF THESE CURVES(LINEAR INTERPOLATE) AND SUBSEQUENTLY
C       INTERPOLATE THESE VALUES OVER THE CURRENT NORMAL STRESSES,
```

```
C       USING DEVIDED DIFFERENCES.
C
      INDEX = 0
      DO 100 I = 1, 18
          IF (DABS(GAMCRK).GE.A(I) .AND. DABS(GAMCRK).LT.A(I+1)) THEN
              INDEX = I
              GO TO 110
          ENDIF
  100 CONTINUE
C
      IF (DABS(GAMCRK) .GE. A(19)) THEN
          TAU12C = B(19)
          TAU34C = C(19)
          TAU56C = D(19)
          TAU78C = E(19)
          GO TO 120
      ENDIF
  110 CONTINUE
      GAM1 = A(INDEX)
      GAM2 = A(INDEX+1)
      TAU1 = B(INDEX)
      TAU2 = B(INDEX+1)
      TAU3 = C(INDEX)
      TAU4 = C(INDEX+1)
      TAU5 = D(INDEX)
      TAU6 = D(INDEX+1)
      TAU7 = E(INDEX)
      TAU8 = E(INDEX+1)
C
      TAU12C = TAU1 + (TAU2-TAU1)/(GAM2-GAM1)*(DABS(GAMCRK)-GAM1)
      TAU34C = TAU3 + (TAU4-TAU3)/(GAM2-GAM1)*(DABS(GAMCRK)-GAM1)
      TAU56C = TAU5 + (TAU6-TAU5)/(GAM2-GAM1)*(DABS(GAMCRK)-GAM1)
      TAU78C = TAU7 + (TAU8-TAU7)/(GAM2-GAM1)*(DABS(GAMCRK)-GAM1)
C
C  USING DEVIDED DIFFERENCES FIND CORRECT TAU FOR CURRENT SIGNORMAL
C
  120 CONTINUE
      FX0 = TAU12C
      FX01 = (TAU34C-TAU12C)/0.50
      FX12 = (TAU56C-TAU34C)/0.50
      FX34 = (TAU78C-TAU56C)/1.0
      FX012 = FX12 - FX01
      FX123 = (FX34-FX12)/1.5
      FX0123 = (FX123-FX012)/2.0
C
      TAUCRK = FX0 + FX01*SIGBND + FX012*(SIGBND-0.50)*SIGBND + FX0123*(
     1    SIGBND-1.0)*(SIGBND-0.5)*SIGBND
C
C       CRACK SHEAR STIFFNESS IS ...
C
      GCRACK = DABS(TAUCRK/GAMCRK)
C
      RETURN
      END


C ====================== D C O N S T ==============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE DCONST(ELNUM,ITYPE,MATNUM,INTGPN,IFLAG,IOUT
     $                 ,DETJAC,ICODE,NNEL)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
```

```
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C DMATCS    CSSTSN
C
C************************************************************
      INTEGER ELNUM
C
      IF (ICODE .EQ. 0) THEN
          IPG = 0
          CALL DMATCS (ELNUM, ITYPE, INTGPN, IFLAG, IOUT, ICODE, IPG)
      ELSE
          CALL CSSTSN (ELNUM, ITYPE, INTGPN, IFLAG, IOUT, NNEL, ICODE)
      ENDIF
      RETURN
      END
C
C ======================= C S S T S N ================================
C
C     INCLUDE (PROCESS)
      SUBROUTINE CSSTSN(ELNUM,ITYPE,INTGPN,IFLAG,IOUT,NNEL,ICODE)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C IOGET     DMATCS    IOPUT
C
C************************************************************
      CHARACTER*105 DUMMY
      REAL*8 LTHICK
      INTEGER ELNUM
      COMMON/UTIL1/STRESS(6),STRAIN(6),DUMMY
      COMMON/MATER1/DEP(6,6)
      COMMON/ELSTR1/STRN(6)
      COMMON/ELSTR2/STRS(6)
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/CONTR1/INCREM,NIT
      COMMON/IALGTM/IFLAG5
      COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
      COMMON/LPROP/PROPER(25,15)
      DIMENSION DE(6),DS(6)
C
      IF (ITYPE .GT. 300) THEN
          IEND = 6
      ELSE
          IEND = 4
      ENDIF
C
      IF (INCREM.GT.1 .OR. NIT.GT.1) THEN
          CALL IOGET (LDEV1, 96, '(A96)', 5)
      ELSE
          DO 100 K1 = 1, IEND
              STRESS(K1) = 0.
              STRAIN(K1) = 0.
  100     CONTINUE
      ENDIF
C
      IF (IFLAG5 .EQ. 1) THEN
          DO 110 K1 = 1, IEND
```

```
            DE(K1) = STRN(K1) - STRAIN(K1)
  110     CONTINUE
      ELSE IF (IFLAG5 .EQ. 0) THEN
          DO 120 K1 = 1, IEND
              DE(K1) = STRN(K1)
  120     CONTINUE
      ENDIF
C
      IPG = 0
      CALL DMATCS (ELNUM, ITYPE, INTGPN, IFLAG, IOUT, ICODE, IPG)
C
      DO 140 K1 = 1, IEND
          S = 0.
          DO 130 K2 = 1, IEND
              S = S + DEP(K1,K2)*DE(K2)
  130     CONTINUE
          DS(K1) = S
  140 CONTINUE
C
      DO 150 K1 = 1, IEND
          STRAIN(K1) = STRN(K1)
          IF (IFLAG5 .EQ. 1) THEN
              STRESS(K1) = STRESS(K1) + DS(K1)
          ELSE IF (IFLAG5 .EQ. 0) THEN
              STRESS(K1) = DS(K1)
          ENDIF
          STRS(K1) = STRESS(K1)
  150 CONTINUE
      IF (IFLAG5 .EQ. 1) THEN
          CALL IOPUT (LDEV2, 96, '(A96)', 5)
      ELSE IF (IFLAG5 .EQ. 0) THEN
          CALL IOPUT (LDEV3, 96, '(A96)', 5)
      ENDIF
      RETURN
  900 FORMAT(2A48)
      END
C
C ===================== D M A T C S ================================
C
C     INCLUDE (PROCESS)
      SUBROUTINE DMATCS(ELNUM,ITYPE,INTGPN,IFLAG,IOUT,ICODE,IPG)
C
C =================================================================
C I
C I   PROGRAM 'DMATCS' EVALUATES THE STRESS-STRAIN STIFFNESS MATRIX
C I   FOR NONLINEAR INELASTIC MATERIALS
C I
C I   C O M M O N     B L O C K S
C I
C I
C ================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C DSHELL
C
C************************************************************
      REAL*8 LTHICK
```

```
      INTEGER ELNUM
      COMMON/LPROP/PROPER(25,15)
      COMMON/MATER1/DEP(6,6)
      COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      IF (ITYPE .GT. 300) THEN
C
          DO 110 K2 = 1, 6
              DO 100 K1 = 1, 6
                  DEP(K1,K2) = 0.
  100         CONTINUE
  110     CONTINUE
C
          DEP(1,1) = CRACK(INTGPN,2,LYNUM)
          DEP(1,2) = CRACK(INTGPN,3,LYNUM)
          DEP(1,3) = CRACK(INTGPN,4,LYNUM)
          DEP(1,4) = CRACK(INTGPN,5,LYNUM)
          DEP(1,5) = CRACK(INTGPN,6,LYNUM)
          DEP(1,6) = CRACK(INTGPN,7,LYNUM)
          DEP(2,1) = CRACK(INTGPN,8,LYNUM)
          DEP(2,2) = CRACK(INTGPN,9,LYNUM)
          DEP(2,3) = CRACK(INTGPN,10,LYNUM)
          DEP(2,4) = CRACK(INTGPN,11,LYNUM)
          DEP(2,5) = CRACK(INTGPN,12,LYNUM)
          DEP(2,6) = CRACK(INTGPN,13,LYNUM)
          DEP(3,1) = CRACK(INTGPN,14,LYNUM)
          DEP(3,2) = CRACK(INTGPN,15,LYNUM)
          DEP(3,3) = CRACK(INTGPN,16,LYNUM)
          DEP(3,4) = CRACK(INTGPN,17,LYNUM)
          DEP(3,5) = CRACK(INTGPN,18,LYNUM)
          DEP(3,6) = CRACK(INTGPN,19,LYNUM)
          DEP(4,1) = CRACK(INTGPN,20,LYNUM)
          DEP(4,2) = CRACK(INTGPN,21,LYNUM)
          DEP(4,3) = CRACK(INTGPN,22,LYNUM)
          DEP(4,4) = CRACK(INTGPN,23,LYNUM)
          DEP(4,5) = CRACK(INTGPN,24,LYNUM)
          DEP(4,6) = CRACK(INTGPN,25,LYNUM)
          DEP(5,1) = CRACK(INTGPN,26,LYNUM)
          DEP(5,2) = CRACK(INTGPN,27,LYNUM)
          DEP(5,3) = CRACK(INTGPN,28,LYNUM)
          DEP(5,4) = CRACK(INTGPN,29,LYNUM)
          DEP(5,5) = CRACK(INTGPN,30,LYNUM)
          DEP(5,6) = CRACK(INTGPN,31,LYNUM)
          DEP(6,1) = CRACK(INTGPN,32,LYNUM)
          DEP(6,2) = CRACK(INTGPN,33,LYNUM)
          DEP(6,3) = CRACK(INTGPN,34,LYNUM)
          DEP(6,4) = CRACK(INTGPN,35,LYNUM)
          DEP(6,5) = CRACK(INTGPN,36,LYNUM)

          DEP(6,6) = CRACK(INTGPN,37,LYNUM)
C
C
C ----- solid brick element recover the global D matrix directly and
C       store it in DEP as above. The shell element is transformed to
C       local and reduced in the shell normal direction and transformed
C       back to the global direction.
C
          IF (IFLAG .EQ. 4) THEN
              IF (PROPER(25,MATRL) .NE. 0.0) CALL DSHELL (IOUT, ICODE,
     1            ELNUM, INTGPN, IPG)
          ENDIF
      ENDIF
```

```
      RETURN
      END
C=================================================================
C     INCLUDE (PROCESS)
      SUBROUTINE DSHELL(IOUT,ICODE,ELNUM,INTGPN,IPG)
C
C ==================================================================
C I
C I   PROGRAM 'DMATCS' EVALUATES THE STRESS-STRAIN STIFFNESS MATRIX
C I   FOR NONLINEAR INELASTIC MATERIALS
C I
C I   C O M M O N     B L O C K S
C I
C I
C =================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C**************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C SIMUL
C
C**************************************************************
      INTEGER ELNUM
      REAL*8 LTHICK
      COMMON/MATER1/DEP(6,6)
      COMMON/TRANS/V1(3),V2(3),V3(3)
      COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/TSHEAR/AK1(5000,9),AK2(5000,9),AF1(5000,15,27),
     $              AF2(5000,15,27)
      DIMENSION TEMP(6,6),DC(6,6),TEMP3(6,6)
C
      IPOINT = INTGPN/(NIPXI*NIPETA)
      INDEX = INTGPN - IPOINT*NIPXI*NIPETA
      IF (INDEX .EQ. 0) INDEX = NIPXI*NIPETA
C
      DC(1,1) = V1(1)**2
      DC(2,1) = V2(1)**2
      DC(3,1) = V3(1)**2
      DC(1,2) = V1(2)**2
      DC(2,2) = V2(2)**2
      DC(3,2) = V3(2)**2
      DC(1,3) = V1(3)**2
      DC(2,3) = V2(3)**2
      DC(3,3) = V3(3)**2
      DC(4,1) = 2.*V1(1)*V2(1)
      DC(5,1) = 2.*V2(1)*V3(1)
      DC(6,1) = 2.*V1(1)*V3(1)
      DC(4,2) = 2.*V1(2)*V2(2)
      DC(5,2) = 2.*V2(2)*V3(2)
      DC(6,2) = 2.*V1(2)*V3(2)
      DC(4,3) = 2.*V1(3)*V2(3)
      DC(5,3) = 2.*V2(3)*V3(3)
      DC(6,3) = 2.*V1(3)*V3(3)
      DC(1,4) = V1(1)*V1(2)
      DC(2,4) = V2(1)*V2(2)
      DC(3,4) = V3(1)*V3(2)
      DC(1,5) = V1(2)*V1(3)
      DC(2,5) = V2(2)*V2(3)
```

```
      DC(3,5) = V3(2)*V3(3)
      DC(1,6) = V1(1)*V1(3)
      DC(2,6) = V2(1)*V2(3)
      DC(3,6) = V3(1)*V3(3)
      DC(4,4) = V1(1)*V2(2) + V2(1)*V1(2)
      DC(5,4) = V2(1)*V3(2) + V3(1)*V2(2)
      DC(6,4) = V3(1)*V1(2) + V1(1)*V3(2)
      DC(4,5) = V1(3)*V2(2) + V2(3)*V1(2)
      DC(5,5) = V2(3)*V3(2) + V3(3)*V2(2)
      DC(6,5) = V3(3)*V1(2) + V1(3)*V3(2)
      DC(4,6) = V1(3)*V2(1) + V2(3)*V1(1)
      DC(5,6) = V2(3)*V3(1) + V3(3)*V2(1)
      DC(6,6) = V3(3)*V1(1) + V1(3)*V3(1)
C
C ----- TRANSFORM DEP FROM GLOBAL TO LOCAL COORDINATES BY USING THE
C       transformation matrix DC.
C
      DO 110 K1 = 1, 6
          DO 100 K2 = 1, 6
              TEMP3(K1,K2) = DC(K1,K2)
  100     CONTINUE
  110 CONTINUE
      N = 6
      EPS = 0.0001
      NN = 6
      DETER = SIMUL(N,TEMP3,EPS,NN,IOUT)
      IF (DETER .EQ. 0.0) THEN
          WRITE (*, *) 'ERROR IN SUB ELASTIC---SIMUL INVERSE'
          STOP
      ENDIF
C
      DO 140 K1 = 1, 6
          DO 130 K2 = 1, 6
              TEMP1 = 0.
              DO 120 K3 = 1, 6
                  TEMP1 = TEMP1 + TEMP3(K3,K1)*DEP(K3,K2)
  120         CONTINUE
              TEMP(K1,K2) = TEMP1
  130     CONTINUE
  140 CONTINUE
C
      DO 170 K1 = 1, 6
          DO 160 K2 = 1, 6
              TEMP1 = 0.
              DO 150 K3 = 1, 6
                  TEMP1 = TEMP1 + TEMP(K1,K3)*TEMP3(K3,K2)
  150         CONTINUE
              DEP(K1,K2) = TEMP1
  160     CONTINUE
  170 CONTINUE
C
C ----  The shell normal stiffness(local) is suppressed here
C
      DO 210 I = 1, 6
          IF (I .EQ. 3) GO TO 200
          DO 190 J = 1, 6
              IF (J .EQ. 3) GO TO 180
              DEP(I,J) = DEP(I,J) - DEP(3,J)*DEP(I,3)/DEP(3,3)
  180         CONTINUE
  190     CONTINUE
  200     CONTINUE
  210 CONTINUE
      DO K1 = 1, 6
```

```
          DEP(3,K1) = 0.0
          DEP(K1,3) = 0.0
      END DO
C
C ... SHEAR CORRECTION FACTOR FOR SHELLS
C
      AFACT2 = AF1(ELNUM,LYNUM,INTGPN)*5./6.
      AFACT1 = AF2(ELNUM,LYNUM,INTGPN)*5./6.
C
      IF (NLAYRS .EQ. 1) THEN
          AFACT1 = 5./6.
          AFACT2 = 5./6.
      ENDIF
c
      DEP(5,5) = DEP(5,5)*AFACT1
      DEP(6,6) = DEP(6,6)*AFACT2
c
      IF (IPG .EQ. 1) RETURN
C
C ----- TRANSFORM DEP TO THE GLOBAL COORDINATES BY USING THE
C         transformation matrix <DC.
c
      DO 250 K1 = 1, 6
          DO 240 K2 = 1, 6
              TEMP1 = 0.
              DO 230 K3 = 1, 6
                  TEMP1 = TEMP1 + DC(K3,K1)*DEP(K3,K2)
  230         CONTINUE
              TEMP(K1,K2) = TEMP1
  240     CONTINUE
  250 CONTINUE
C
      DO 280 K1 = 1, 6
          DO 270 K2 = 1, 6
              TEMP1 = 0.
              DO 260 K3 = 1, 6
                  TEMP1 = TEMP1 + TEMP(K1,K3)*DC(K3,K2)
  260         CONTINUE
              DEP(K1,K2) = TEMP1
  270     CONTINUE
  280 CONTINUE
C
      RETURN
      END


C
C ========================= G A U S S =================================
C
C     INCLUDE (PROCESS)
      SUBROUTINE GAUSS(NIP,W,GCOORD)
C
C ====================================================================
C I
C I      SUBPROGRAM GAUSS STORES THE COORDINATES XI AND ETA OF THE
C I      NUMERICAL INTEGRATION POINTS AND THEIR WEIGHTING FUNCTIONS
C I      FOR THE FOUR POINT AND THE NINE POINT INTEGRATION.
C I
C I          NIP       = NUMBER OF THE INTEGRATION POINTS
C I          W(I)      = WEIGTH FUNCTION
C I          GCOORD(I) = COORDINATES OF THE GAUSSIAN POINTS
C I                      FROM THE NEGATIVE TO POSITIVE
C I
C ===================================================================
```

```
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C************************************************************
      REAL*8 W,GCOORD
      DIMENSION W(4),GCOORD(4)
C
      IF (NIP .EQ. 1) THEN
          W(1) = 2.0D0
          GCOORD(1) = 0.0D0
C
      ELSE IF (NIP .EQ. 2) THEN
          W(1) = 1.0D0
          W(2) = 1.0D0
          GCOORD(1) = -0.577350269189626D0
          GCOORD(2) = 0.577350269189626D0
C
      ELSE IF (NIP .EQ. 3) THEN
          W(1) = 5.0D0/9.0D0
          W(2) = 8.0D0/9.0D0
          W(3) = W(1)
          GCOORD(1) = -0.774596669241483D0
          GCOORD(2) = 0.
          GCOORD(3) = 0.774596669241483D0
C
      ELSE IF (NIP .EQ. 4) THEN
          W(1) = 0.347854845137454D0
          W(2) = 0.652145154862546D0
          W(3) = W(2)
          W(4) = W(1)
          GCOORD(1) = -0.861136311594053D0
          GCOORD(2) = -0.339981043584856D0
          GCOORD(3) = 0.339981043584856D0
          GCOORD(4) = 0.861136311594053D0
C
      ELSE
          GO TO 100
C
      ENDIF
  100 CONTINUE
      RETURN
      END
C
C
C ====================== I S H A P E ===============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE ISHAPE
C =================================================================
C I
C I   THIS PROGRAM EVALUATES THE SHAPE FUNCTIONS, THEIR DERIVATIVES
C I   WITH RESPECT TO THE NATURAL COORDINATES, AND THE WEIGHT
C I   FUNCTIONS AT EACH INTEGRATION POINT.
C I
C I   ENTRY POINTS:
C I       ISH2DG     (FOR 2D ELEMENTS)
C I       ISH3DG     (FOR 3D ELEMENTS)
C I
```

```
C I
C ==================================================================
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C GAUSS     N2D       ISOP2D    ISHEXT    ISOP3D
C GAUSSL
C
C************************************************************
      REAL*8 N,NXI,NETA,NSI,F,FXI,FETA,FSI,ASI,SI,LTHICK
      COMMON/ISHAP1/N(20,27),NXI(20,27),NETA(20,27),NSI(20,27),SI(27)
      COMMON/ISHAP2/W(27)
      COMMON/CONSIS/RSTLOC(27,3)
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
      DIMENSION F(20),FXI(20),FETA(20),FSI(20)
      DIMENSION WXI(4),WETA(4),WSI(4),XI(4),ETA(4),SIP(4)
C
C ----- EVALUATE THE SHAPE FUNCTIONS AND THEIR DERIVATIVES WITH RESPECT
C       TO THE ISOPARAMETRIC COORDINATES FOR TWO DIMENSIONAL ELEMENTS
C       AT EACH GAUSSIAN INTEGRATION POINT.
C
      ENTRY ISH2DG(ITYPE,NNEL,IERROR)
C
C ----- GET THE ISOPARAMETRIC COORDINATES OF INTEGRATION POINTS
C
      CALL GAUSS (NIPXI, WXI, XI)
      CALL GAUSS (NIPETA, WETA, ETA)
      NIP = NIPXI*NIPETA
C
C ----- NIP = TOTAL NUMBER OF INTEGRATION POINTS FOR THE ELEMENT
C ----- IETA = ROW NUMBER OF THE GAUSSIAN POINT FROM THE BOTTOM
C ----- IXI = COLUMN NUMBER OF THE GAUSSIAN POINT FROM LEFT
C
      DO 110 IETA = 1, NIPETA
          DO 100 IXI = 1, NIPXI
C
              J = (IETA-1)*NIPXI + IXI
              W(J) = WXI(IXI)*WETA(IETA)
              AXI = XI(IXI)
              AETA = ETA(IETA)
              CALL N2D (AXI, AETA, F, ITYPE, IERROR)
              CALL ISOP2D (AXI, AETA, FXI, FETA, ITYPE, IERROR)
              CALL ISHEXT (NNEL, J, F, FXI, FETA, FSI, ASI)
  100     CONTINUE
  110 CONTINUE
C
      RETURN
C
C
C
C
C
C ----- EVALUATE THE SHAPE FUNCTIONS AND THEIR DERIVATIVES WITH RESPECT
C       TO THE ISOPARAMETRIC COORDINATES FOR THREE DIMENSIONAL ELEMENTS
C       AT EACH GAUSSIAN INTEGRATION POINT.
C
      ENTRY ISH3DG (ITYPE, NNEL, IERROR)
C
```

```
      CALL GAUSS (NIPXI, WXI, XI)
      CALL GAUSS (NIPETA, WETA, ETA)
      CALL GAUSS (NIPSI, WSI, SIP)
C     WRITE(*,*) 'ISH3DG ENTERED'
      NIP = NIPXI*NIPETA*NIPSI
      I = NIPXI*NIPETA
C
      DO 140 ISI = 1, NIPSI
          DO 130 IETA = 1, NIPETA
              DO 120 IXI = 1, NIPXI
                  J = (ISI-1)*I + (IETA-1)*NIPXI + IXI
                  W(J) = WXI(IXI)*WETA(IETA)*WSI(ISI)
                  AXI = XI(IXI)
                  AETA = ETA(IETA)
                  ASI = SIP(ISI)
C****
                  RSTLOC(J,1) = AXI
                  RSTLOC(J,2) = AETA
                  RSTLOC(J,3) = ASI
C****


                  CALL ISOP3D (AXI, AETA, ASI, F, FXI, FETA, FSI, ITYPE
     1                  , IERROR)
                  CALL ISHEXT (NNEL, J, F, FXI, FETA, FSI, ASI)
  120         CONTINUE
  130     CONTINUE
  140 CONTINUE
C
      RETURN
C
C ----- EVALUATE THE SHAPE FUNCTIONS AND THEIR DERIVATIVES WITH RESPECT
C       TO THE ISOPARAMETRIC COORDINATES FOR THREE DIMENSIONAL SHELL
C       ELEMENTS AT EACH GAUSSIAN INTEGRATION POINT.
C
C
      ENTRY ISHSHL (ITYPE, NNEL, IERROR)
C
      IT = ITYPE - 100
      CALL GAUSS (NIPXI, WXI, XI)
      CALL GAUSS (NIPETA, WETA, ETA)
      IF (NLAYRS .EQ. 1) CALL GAUSS (NIPSI, WSI, SIP)
      NIP = NIPXI*NIPETA*NIPSI
      I = NIPXI*NIPETA
C
C
      DO 170 K1 = 1, 20
          FSI(K1) = 0.0D0
  170 CONTINUE
C
      DO 200 ISI = 1, NIPSI
          DO 190 IETA = 1, NIPETA
              DO 180 IXI = 1, NIPXI
                  J = (ISI-1)*I + (IETA-1)*NIPXI + IXI
                  IF (NLAYRS .EQ. 1) THEN
                      W(J) = WXI(IXI)*WETA(IETA)*WSI(ISI)
                      ASI = SIP(ISI)
                  ENDIF
                  AXI = XI(IXI)
                  AETA = ETA(IETA)
                  CALL N2D (AXI, AETA, F, IT, IERROR)
                  CALL ISOP2D (AXI, AETA, FXI, FETA, IT, IERROR)
                  IF (NLAYRS .GT. 1) THEN
                      CALL GAUSSL (F, SIP, WSI, NNEL, ISI, NIPSI)
```

```
                     ASI = SIP(ISI)
                     W(J) = WXI(IXI)*WETA(IETA)*WSI(ISI)
                  ENDIF
                  RSTLOC(J,1) = AXI
                  RSTLOC(J,2) = AETA
                  RSTLOC(J,3) = ASI
                  CALL ISHEXT (NNEL, J, F, FXI, FETA, FSI, ASI)
 180          CONTINUE
 190      CONTINUE
 200  CONTINUE
C
      RETURN
      END
C
C ============================ G A U S S L==========================
C
C
C     INCLUDE(PROCESS)
      SUBROUTINE GAUSSL(F,SIP,WSI,NNEL,ISI,NIPSI)
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C GETLTK
C
C***********************************************************
      REAL*8 LTKISI
      DIMENSION F(20),WSI(4),SIP(4)
      CALL GETLTK (NNEL, THICKE, ZSI, F, THICKL)
C
      ZIISI = ZSI*2.0/THICKE
      LTKISI = THICKL*2.0/THICKE
      IF (NIPSI .EQ. 1) THEN
          IF (ISI .EQ. 1) THEN
              SIP(ISI) = ZIISI
              WSI(ISI) = 2.0*THICKL/THICKE
          ENDIF
      ELSE IF (NIPSI .EQ. 2) THEN
          IF (ISI .EQ. 1) THEN
              SIP(ISI) = ZIISI - (LTKISI/2.0)*0.57735
              WSI(ISI) = THICKL/THICKE
          ELSE IF (ISI .EQ. 2) THEN
              SIP(ISI) = ZIISI + (LTKISI/2.0)*0.57735
              WSI(ISI) = THICKL/THICKE
          ENDIF
      ELSE IF (NIPSI .EQ. 3) THEN
          IF (ISI .EQ. 1) THEN
              SIP(ISI) = ZIISI - (LTKISI/2.0)*0.774596669241483
              WSI(ISI) = (THICKL/THICKE)*5.0/9.0
          ELSE IF (ISI .EQ. 2) THEN
              SIP(ISI) = ZIISI
              WSI(ISI) = (THICKL/THICKE)*8.0/9.0
          ELSE IF (ISI .EQ. 3) THEN
              SIP(ISI) = ZIISI + (LTKISI/2.0)*0.774596669241483
              WSI(ISI) = (THICKL/THICKE)*5.0/9.0
          ENDIF
      ENDIF
      RETURN
      END
```

```
C =====================G E T L A Y E R T H I C K=====================
C
C      INCLUDE(PROCESS)
       SUBROUTINE GETLTK(NNEL,THICKE,ZSI,F,THICKL)
C
       IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C***********************************************************
       REAL*4 THICK
       REAL*8 LTHICK
       COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
       COMMON/INPUT9/THICK(9),IFLAG
       DIMENSION F(20)
C
       THICKE = 0.0
       THICKL = 0.0
       ZSI = 0.0
       DO 100 K = 1, NNEL
           ZSI = ZSI + F(K)*ZS(K)
           THICKL = THICKL + LTHICK(K)*F(K)
           THICKE = THICKE + THICK(K)*F(K)
  100 CONTINUE
       RETURN
       END
C ========================= I S H E X T ===================
C
C
C      INCLUDE (PROCESS)
       SUBROUTINE ISHEXT(NNEL,J,F,FXI,FETA,FSI,ASI)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C***********************************************************
       REAL*8 N,NXI,NETA,NSI,F,FXI,FETA,FSI,ASI,SI
       COMMON/ISHAP1/N(20,27),NXI(20,27),NETA(20,27),NSI(20,27),SI(27)
       DIMENSION F(20),FXI(20),FETA(20),FSI(20)
C
C*VDIR: ASSUME COUNT(8)
C
       DO 100 NN = 1, NNEL
           N(NN,J) = F(NN)
           NXI(NN,J) = FXI(NN)
           NETA(NN,J) = FETA(NN)
           NSI(NN,J) = FSI(NN)
           SI(J) = ASI
  100 CONTINUE
C
C       DO 99 IMN=1,NNEL
C       WRITE(*,*)  'INTGPN=',J,'IMN=',IMN, 'ISHEXT N',  N(IMN,J)
C99     CONTINUE
       RETURN
```

```
      END
C
C ========================= E L M L I B ===========================
C
C     INCLUDE (PROCESS)
      SUBROUTINE ELMLIB
C
C =================================================================
C I
C I      SUBPROGRAM ELMLIB CALCULATES THE SHAPE FUNCTIONS AND THE
C I      PARTIAL DRIVATIVES OF THE SHAPE FUNCTIONS WRT THE LOCAL
C I      COORDINATES 'XI', 'ETA' AND 'SI'.
C I
C I              N(I)     = SHAPE FUNCTIONS OF THE ELEMENT
C I              NXI(I) = PARTIAL DRIVATIVE OF 'N' WRT 'XI'
C I              NETA(I) =    PARTIAL DRIVATIVE OF 'N' WRT 'ETA'
C I              NSI(I) = PARTIAL DERIVATIVE OF 'N' WRT 'SI'
C I
C I
C I
C ================================================================
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ELMEXT
C
C***********************************************************
      REAL*8 XI,ETA,SI,N,NXI,NETA,NSI,XIO,ETAO,SIO
      COMMON/ELLIB1/XII(20),ETAI(20),SII(20)
      DIMENSION N(20),NXI(20),NETA(20),NSI(20)
C
C
C
      ENTRY ISOP2D(XI,ETA,NXI,NETA,ITYPE,IERROR)
C
C ---- DRIVATIVE OF SHAPE FUNCTIONS FOR 2D ISOPARAMETIC QUADRILATERAL
C        ELEMENTS
C
C
      NXI(1) = -0.25*(1.-ETA)
      NXI(2) = 0.25*(1.-ETA)
      NXI(3) = 0.25*(1.+ETA)
      NXI(4) = -0.25*(1.+ETA)
      NETA(1) = -0.25*(1.-XI)
      NETA(2) = -0.25*(1.+XI)
      NETA(3) = 0.25*(1.+XI)
      NETA(4) = 0.25*(1.-XI)
C
      IF (ITYPE .EQ. 219) THEN
C
C ---- DRIVATIVE OF SHAPE FUNCTIONS FOR 9 NODES SERENDIPITY
C        ISOPARAMETRIC ELEMENT.
C
          CST = 9.D0/32.D0
          C13 = 1.D0/3.D0
          C23 = 2.D0/3.D0
          NXI(5) = CST*(1.-ETA)*(9.*XI**2-2.*XI-3.)
          NXI(6) = CST*(1.-ETA)*((-9.*XI**2)-2.*XI+3.)
          NXI(7) = 0.5*(1.-ETA**2)
          NXI(8) = -XI*(1+ETA)
```

```
          NXI(9) = -0.5*(1.-ETA**2)
          NXI(1) = NXI(1) - 0.5*NXI(9) - C23*NXI(5) - C13*NXI(6)
          NXI(2) = NXI(2) - 0.5*NXI(7) - C23*NXI(6) - C13*NXI(5)
          NXI(3) = NXI(3) - 0.5*(NXI(7)+NXI(8))
          NXI(4) = NXI(4) - 0.5*(NXI(8)+NXI(9))
          NETA(5) = -CST*(1.-XI**2)*(1.-3.*XI)
          NETA(6) = -CST*(1.-XI**2)*(1.+3.*XI)
          NETA(7) = -ETA*(1.+XI)
          NETA(8) = 0.5*(1.-XI**2)
          NETA(9) = -ETA*(1.-XI)
          NETA(1) = NETA(1) - 0.5*NETA(9) - C23*NETA(5) - C13*NETA(6)
          NETA(2) = NETA(2) - 0.5*NETA(7) - C23*NETA(6) - C13*NETA(5)
          NETA(3) = NETA(3) - 0.5*(NETA(7)+NETA(8))
          NETA(4) = NETA(4) - 0.5*(NETA(8)+NETA(9))
      ELSE
          IF (ITYPE .EQ. 204) GO TO 100
C
C ---- ADDITIONAL TERMS FOR THE FIVE NODE ISOPARAMETRIC QUADRILATERAL
C       ELEMENT.
C
          NXI(5) = -XI*(1.-ETA)
          NETA(5) = -0.5*(1.-XI**2)
          NXI(1) = NXI(1) - 0.5*NXI(5)
          NXI(2) = NXI(2) - 0.5*NXI(5)
          NETA(1) = NETA(1) - 0.5*NETA(5)
          NETA(2) = NETA(2) - 0.5*NETA(5)
C
          IF (ITYPE .EQ. 205) GO TO 100
C
C ---- ADDITIONAL TERMS FOR THE EIGHT NODE ISOPARAMETRIC QUADRILATERAL
C       ELEMENT.
C
          NXI(6) = 0.5*(1.-ETA**2)
          NXI(7) = -XI*(1+ETA)
          NXI(8) = -0.50*(1.-ETA**2)
          NETA(6) = -(1.+XI)*ETA
          NETA(7) = 0.5*(1.-XI**2)
          NETA(8) = -(1.-XI)*ETA
          NXI(1) = NXI(1) - 0.5*NXI(8)
          NXI(2) = NXI(2) - 0.5*NXI(6)
          NXI(3) = NXI(3) - 0.5*(NXI(7)+NXI(6))
          NXI(4) = NXI(4) - 0.5*(NXI(7)+NXI(8))
          NETA(1) = NETA(1) - 0.5*NETA(8)
          NETA(2) = NETA(2) - 0.5*NETA(6)
          NETA(3) = NETA(3) - 0.5*(NETA(7)+NETA(6))
          NETA(4) = NETA(4) - 0.5*(NETA(7)+NETA(8))
C
          IF (ITYPE .EQ. 208) GO TO 100
C
C ---- ADDITIONAL TERMS FOR THE NINE NODE LAGRANGIAN ISOPARAMETRIC
C       QUADRILATERAL ELEMENT.
C
          NXI(9) = -2.*XI*(1.-ETA**2)
          NXI(1) = NXI(1) + NXI(9)/4.
          NXI(2) = NXI(2) + NXI(9)/4.
          NXI(3) = NXI(3) + NXI(9)/4.
          NXI(4) = NXI(4) + NXI(9)/4.
          NXI(5) = NXI(5) - NXI(9)/2.
          NXI(6) = NXI(6) - NXI(9)/2.
          NXI(7) = NXI(7) - NXI(9)/2.
          NXI(8) = NXI(8) - NXI(9)/2.
          NETA(9) = -2.*ETA*(1.-XI**2)
          NETA(1) = NETA(1) + NETA(9)/4.
```

```
         NETA(2) = NETA(2) + NETA(9)/4.
         NETA(3) = NETA(3) + NETA(9)/4.
         NETA(4) = NETA(4) + NETA(9)/4.
         NETA(5) = NETA(5) - NETA(9)/2.
         NETA(6) = NETA(6) - NETA(9)/2.
         NETA(7) = NETA(7) - NETA(9)/2.
         NETA(8) = NETA(8) - NETA(9)/2.
      ENDIF
  100 CONTINUE
      RETURN
C
C
C
      ENTRY N2D (XI, ETA, N, ITYPE, IERROR)
C
C ---- SHAPE FUNCTIONS FOR 2D QUADRILATERAL ISOPARAMETRIC ELEMENTS.
C
C
      N(1) = 0.25*(1.-XI)*(1.-ETA)
      N(2) = 0.25*(1.+XI)*(1.-ETA)
      N(3) = 0.25*(1.+XI)*(1.+ETA)
      N(4) = 0.25*(1.-XI)*(1.+ETA)
C
      IF (ITYPE .EQ. 219) THEN
C
C ---- SHAPE FUNCTIONS FOR 9 NODES SERENDIPITY
C        ISOPARAMETRIC ELEMENT.
C
         CST = 9.D0/32.D0
         C13 = 1.D0/3.D0
         C23 = 2.D0/3.D0
         N(5) = CST*(1.-ETA)*(1.-XI**2)*(1.-3.*XI)
         N(6) = CST*(1.-ETA)*(1.-XI**2)*(1.+3.*XI)
         N(7) = 0.5*(1.+XI)*(1.-ETA**2)
         N(8) = 0.5*(1.-XI**2)*(1+ETA)
         N(9) = 0.5*(1.-XI)*(1.-ETA**2)
         N(1) = N(1) - 0.5*N(9) - C23*N(5) - C13*N(6)
         N(2) = N(2) - 0.5*N(7) - C23*N(6) - C13*N(5)
         N(3) = N(3) - 0.5*(N(7)+N(8))
         N(4) = N(4) - 0.5*(N(8)+N(9))
      ELSE
         IF (ITYPE .EQ. 204) GO TO 110
C
C ---- ADDITIONAL TERMS FOR THE FIVE NODED SERENDIPITY ELEMENT
C
         N(5) = 0.5*(1.-XI**2)*(1.-ETA)
         N(1) = N(1) - 0.5*N(5)
         N(2) = N(2) - 0.5*N(5)
         IF (ITYPE .EQ. 205) GO TO 110
C
C ---- ADDITIONAL TERMS FOR THE EIGHT NODED SERENDIPITY ELEMENT
C
         N(6) = 0.5*(1.+XI)*(1.-ETA**2)


         N(7) = 0.5*(1.-XI**2)*(1.+ETA)
         N(8) = 0.5*(1.-XI)*(1.-ETA**2)
         N(1) = N(1) - 0.5*N(8)
         N(2) = N(2) - 0.5*N(6)
         N(3) = N(3) - 0.5*(N(7)+N(6))
         N(4) = N(4) - 0.5*(N(7)+N(8))
C
         IF (ITYPE .EQ. 208) GO TO 110
C
```

```
C ---- ADDITIONAL TERMS FOR THE NINE NODED LAGRANGIAN ELEMENT
C
          N(9) = (1.-ETA**2)*(1.-XI**2)
          N(1) = N(1) + N(9)/4.
          N(2) = N(2) + N(9)/4.
          N(3) = N(3) + N(9)/4.
          N(4) = N(4) + N(9)/4.
          N(5) = N(5) - N(9)/2.
          N(6) = N(6) - N(9)/2.
          N(7) = N(7) - N(9)/2.
          N(8) = N(8) - N(9)/2.
      ENDIF
C
  110 CONTINUE
      RETURN
C
C
C
C ---- SHAPE FUNCTIONS AND THEIR DERIVATIVES FOR THE 3D ISOP. EL.
C
      ENTRY ISOP3D (XI, ETA, SI, N, NXI, NETA, NSI, ITYPE, IERROR)
C
C     WRITE(*,*) 'ISOP3D ENTERED'
      IF (ITYPE .EQ. 308) THEN
C
          DO 120 K = 1, 8
              CALL ELMEXT (XI, ETA, SI, K, XIO, ETAO, SIO)
              N(K) = .125*(1.+XIO)*(1.+ETAO)*(1.+SIO)
              NXI(K) = .125*(1.+ETAO)*(1.+SIO)*XII(K)
              NETA(K) = .125*(1.+XIO)*(1.+SIO)*ETAI(K)
              NSI(K) = .125*(1.+XIO)*(1.+ETAO)*SII(K)
  120     CONTINUE
C
      ELSE IF (ITYPE .EQ. 320) THEN
C
C        HEXAHYDRON SOLID ELEMENT
C        SHAPE FUNCTIONS AND THIE DERIVATIVES FOR NODES 1-8.
C
          DO 130 K = 1, 8
              CALL ELMEXT (XI, ETA, SI, K, XIO, ETAO, SIO)
              N(K) = .125*(1.+XIO)*(1.+ETAO)*(1.+SIO)*(XIO+ETAO+SIO-2.)
              NXI(K) = .125*(1.+ETAO)*(1.+SIO)*(2.*XIO+ETAO+SIO-1.)*XII(
     1            K)
              NETA(K) = .125*(1.+XIO)*(1.+SIO)*(XIO+2.*ETAO+SIO-1.)*ETAI
     1            (K)
              NSI(K) = .125*(1.+ETAO)*(1.+XIO)*(XIO+ETAO+2.*SIO-1.)*SII(
     1            K)
  130     CONTINUE
C
          K1 = 9
          K2 = 10
C
C        SHAPE FUNCTIONS AND THEIR DERIVATIVES FOR NODES 13-16.
C
          DO 140 K = 13, 16
              CALL ELMEXT (XI, ETA, SI, K, XIO, ETAO, SIO)
              N(K) = .25*(1.+XIO)*(1.+ETAO)*(1.-SI**2)
              NXI(K) = .25*(1.+ETAO)*(1.-SI**2)*XII(K)
              NETA(K) = .25*(1.+XIO)*(1.-SI**2)*ETAI(K)
              NSI(K) = -0.5*(1.+ETAO)*(1.+XIO)*SI
C
C        SHAPE FUNCTIONS AND THEIR DERIVATIVES FOR NODES 9,11,17,19.
C
```

```
            CALL ELMEXT (XI, ETA, SI, K1, XIO, ETAO, SIO)
            N(K1) = .25*(1.-XI**2)*(1.+ETAO)*(1.+SIO)
            NXI(K1) = -0.5*(1.+ETAO)*(1.+SIO)*XI
            NETA(K1) = .25*(1.-XI**2)*(1.+SIO)*ETAI(K1)
            NSI(K1) = .25*(1.+ETAO)*(1.-XI**2)*SII(K1)
C
C       SHAPE FUNCTIONS AND THEIR DERIVATIVES FOR NODES 10,12,18,20.
C
            CALL ELMEXT (XI, ETA, SI, K2, XIO, ETAO, SIO)
            N(K2) = .25*(1.+XIO)*(1.-ETA**2)*(1.+SIO)
            NXI(K2) = .25*(1.-ETA**2)*(1.+SIO)*XII(K2)
            NETA(K2) = -0.5*(1.+XIO)*(1.+SIO)*ETA
            NSI(K2) = .25*(1.-ETA**2)*(1.+XIO)*SII(K2)
            IF (K1 .EQ. 11) THEN
                K1 = 17
                K2 = 18
            ELSE
                K1 = K1 + 2
                K2 = K2 + 2
            ENDIF
  140     CONTINUE
C
      ENDIF
C
      RETURN
      END
C
C ======================= E L M E X T ============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE ELMEXT(XI,ETA,SI,K,XIO,ETAO,SIO)
C
C ===============================================================
C I
C I
C ===============================================================
C
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C**********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C**********************************************************
      REAL*8 XI,ETA,SI,XIO,ETAO,SIO
      COMMON/ELLIB1/XII(20),ETAI(20),SII(20)
C
C
      XIO = XI*XII(K)
      ETAO = ETA*ETAI(K)
      SIO = SI*SII(K)
C
      RETURN
      END
C
C ======================== J A C O B I ==========================
C
C     INCLUDE (PROCESS)
      SUBROUTINE JACOBI
C
C ===============================================================
```

```
C I
C I        THIS PROGRAM CALCULATES THE JACOBIAN OF THE
C I        TRANSFORMATION BETWEEN THE LOCAL COORDINATES
C I        XI AND ETA AND THE GLOBAL COORDINATES X AND Y
C I        FOR INTEGRATION POINT 'INTGPN' OF ELEMENT NUMBER 'NREL'.
C I
C I   INTGPN  =    INTEGRATION POINT NUMBER
C I   NREL    =    ELEMENT NUMBER
C I   DETJAC  =    DETERMINANT OF THE JACOBIAN
C I   NX(I)   =    PARTIAL DERIVATIVE OF N(I) WITH RESPECT TO X
C I   NY(I)   =    PARTIAL DERIVATIVE OF N(I) WITH RESPECT TO Y
C I   NZ(I)   =    PARTIAL DERIVATIVE OF N(I) WITH RESPECT TO Z
C I   IERROR  =    ERROR CODE FOR EXCESIVE ELEMENT DISTORTION
C I                =1  NO ERROR
C I                =-1 EXCESIVE ELEMENT DISTORTION(DETJAC<0)
C I   NNEL    =    NUMBER OF NODES(SHAPE FUNCTIONS) PER ELEMENT
C I
C I   NXI(I)  =    PARTIAL DERIVATIVE OF N(I) WITH RESPECT TO XI
C I   NETA(I) =    PARTIAL DERIVATIVE OF N(I) WITH RESPECT TO ETA
C I   NSI(I)  =    PARTIAL DERIVATIVE OF N(I) WITH RESPECT TO SI
C I   X(IK)   =    NODE COORDINATE  X,  IK=GLOBAL NODE NUMBER,
C I   Y(IK)   =    NODE COORDINATE  Y,  IK=GLOBAL NODE NUMBER,
C I   Z(IK)   =    NODE COORDINATE  Z,  IK=GLOBAL NODE NUMBER,


C I   XXI     =    PARTIAL DERIVATIVE OF X WITH RESPECT TO XI
C I   XETA    =    PARTIAL DERIVATIVE OF X WITH RESPECT TO ETA
C I   XSI     =    PARTIAL DERIVATIVE OF X WITH RESPECT TO SI
C I   YXI     =    PARTIAL DERIVATIVE OF Y WITH RESPECT TO XI
C I   YETA    =    PARTIAL DERIVATIVE OF Y WITH RESPECT TO ETA
C I   YSI     =    PARTIAL DERIVATIVE OF Y WITH RESPECT TO SI
C I   ZXI     =    PARTIAL DERIVATIVE OF Z WITH RESPECT TO XI
C I   ZETA    =    PARTIAL DERIVATIVE OF Z WITH RESPECT TO ETA
C I   ZSI     =    PARTIAL DERIVATIVE OF Z WITH RESPECT TO SI
C I
C I
C I
C ==================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ERRORS
C
C***********************************************************
      REAL*8 N,NXI,NETA,NSI,NX,NY,NZ
      REAL*4 XYZ
      CHARACTER COMM*6,BUFFER*80,BUFF*80,CELEM*7
      COMMON/ISHAP1/N(20,27),NXI(20,27),NETA(20,27),NSI(20,27),SI(27)
      COMMON/SHLDIR/VECT(3,3,9)
      COMMON/INPUT2/NOP(20,5000)
      COMMON/INPUT3/XYZ(3,10000)
      COMMON/JACOB1/NX(20),NY(20),NZ(20),SIX,SIY,SIZ
      COMMON/COMP2/COMM,BUFFER,BUFF
      DATA ZERO/0.0D0/,CELEM/'ELEMENT'/
      DIMENSION V3( 3 )
      SAVE ZERO,CELEM
C
      ENTRY JACB2D(INTGPN,NREL,NNEL,IERROR,DETJAC)
C
```

```
      XXI = ZERO
      XETA = ZERO
      YXI = ZERO
      YETA = ZERO
C
C*VDIR: ASSUME COUNT(8)
C
      DO 100 K1 = 1, NNEL
          NODE = NOP(K1,NREL)
          X = XYZ(1,NODE)
          Y = XYZ(2,NODE)
          XXI = XXI + NXI(K1,INTGPN)*X
          XETA = XETA + NETA(K1,INTGPN)*X
          YXI = YXI + NXI(K1,INTGPN)*Y
          YETA = YETA + NETA(K1,INTGPN)*Y
  100 CONTINUE
C
      DETJAC = XXI*YETA - YXI*XETA
      DETJ = DETJAC
C
C ---- CHECK THE INTEGRITY OF THE TRANSFORMATION FROM ISOP. COORD.
C       TO THE GLOBAL COORDINATES
C
      IF (DETJAC .EQ. ZERO) THEN
          WRITE (BUFFER, *) CELEM, NREL
          CALL ERRORS (16, 1, 'JACB2D')
C
C ---- DETERMINANT OF JACOBIAN AS EVALUATED ABOVE WILL BE NEGATIVE IF
C       NODE NUMBERING IS DONE IN THE CLOCKWISE DIRECTION. TO HAVE
C       CONSISTENT TRANSFORMATIONS WITH RESPECT TO A 3-D SPACE THE
C       DETERMINANT OF JACOBIAN IS MADE POSITIVE, HOWEVER NX AND NY ARE
C       EVALUATED USING THE NEGATIVE DETERMINANT SAVED IN DETJ VARIABLE.
C
      ELSE IF (DETJAC .LT. ZERO) THEN
          DETJAC = -DETJAC
      ENDIF
C
C*VDIR: ASSUME COUNT(8)
C
      DO 110 K = 1, NNEL
          NX(K) = (YETA*NXI(K,INTGPN)-YXI*NETA(K,INTGPN))/DETJ
          NY(K) = ((-XETA*NXI(K,INTGPN))+XXI*NETA(K,INTGPN))/DETJ
  110 CONTINUE
      RETURN
C
C
      ENTRY JACB3D (INTGPN, NREL, NNEL, IERROR, DETJAC)
C
C     WRITE(*,*) 'JAC3D CALLED'
      XXI = ZERO
      XETA = ZERO
      XSI = ZERO
      YXI = ZERO
      YETA = ZERO
      YSI = ZERO
      ZXI = ZERO
      ZETA = ZERO
      ZSI = ZERO
C
C*VDIR: ASSUME COUNT(20)
C
      DO 120 K1 = 1, NNEL
          NODE = NOP(K1,NREL)
```

```
          X = XYZ(1,NODE)
          Y = XYZ(2,NODE)
          Z = XYZ(3,NODE)
          XXI = XXI + NXI(K1,INTGPN)*X
          XETA = XETA + NETA(K1,INTGPN)*X
          XSI = XSI + NSI(K1,INTGPN)*X
          YXI = YXI + NXI(K1,INTGPN)*Y
          YETA = YETA + NETA(K1,INTGPN)*Y
          YSI = YSI + NSI(K1,INTGPN)*Y
          ZXI = ZXI + NXI(K1,INTGPN)*Z
          ZETA = ZETA + NETA(K1,INTGPN)*Z
          ZSI = ZSI + NSI(K1,INTGPN)*Z
  120 CONTINUE
C
      DETJAC = XXI*(YETA*ZSI-ZETA*YSI) - YXI*(XETA*ZSI-ZETA*XSI) + ZXI*(
     1    XETA*YSI-YETA*XSI)
C
C ---- CHECK THE INTEGRITY OF THE TRANSFORMATION FROM ISOP. COORD.
C       TO THE GLOBAL COORDINATES
C
      IF (DETJAC .EQ. ZERO) THEN
          WRITE (BUFFER, *) CELEM, NREL
          CALL ERRORS (16, 1, 'JACB3D')
      ENDIF
C
C*VDIR: ASSUME COUNT(20)
C
      DO 130 K = 1, NNEL
          NX(K) = ((YETA*ZSI-ZETA*YSI)*NXI(K,INTGPN)-(YXI*ZSI-ZXI*YSI)*
     1        NETA(K,INTGPN)+(YXI*ZETA-ZXI*YETA)*NSI(K,INTGPN))/DETJAC
          NY(K) = ((-(XETA*ZSI-ZETA*XSI)*NXI(K,INTGPN))+(XXI*ZSI-ZXI*XSI
     1        )*NETA(K,INTGPN)-(XXI*ZETA-ZXI*XETA)*NSI(K,INTGPN))/DETJAC
          NZ(K) = ((XETA*YSI-YETA*XSI)*NXI(K,INTGPN)-(XXI*YSI-YXI*XSI)*
     1        NETA(K,INTGPN)+(XXI*YETA-YXI*XETA)*NSI(K,INTGPN))/DETJAC
  130 CONTINUE
      RETURN
C
C
C
C
C
C
C
      ENTRY JACSHL (INTGPN, NREL, NNEL, THICK, DETJAC, V3, ISET, SIP)
C
C     WRITE(*,*) 'INTO JACSHL','INTG.',INTGPN,'NNEL',NNEL,'NREL',NREL,
C    $            'THICK',THICK
      XXI = ZERO
      XETA = ZERO
      XSI = ZERO
      YXI = ZERO
      YETA = ZERO
      YSI = ZERO
      ZXI = ZERO
      ZETA = ZERO
      ZSI = ZERO
C
      XXI1 = ZERO
      XETA1 = ZERO
      YXI1 = ZERO
      YETA1 = ZERO
      ZXI1 = ZERO
      ZETA1 = ZERO
```

```
      VALUE = ZERO
C
C*VDIR: ASSUME COUNT(8)
C
C ----- FIND THE DERIVATIVES OF X,Y AND Z WITH RESPECT TO THE
C       ISOPARAMETRIC COORDINATES. ALSO EVALUATE THE NORMAL TO THE
C       MID PLANE OF SHELL BY INTERPOLATING THE NODAL NORMALS THROUGH
C       THE ELEMENT SHAPE FUNCTIONS. ISET=0
C
C     IF ISET=1 FIND NORMAL AT THE TOP OR BOTTOM SURFACE.
      IF (ISET .EQ. 0) VALUE = 0.0
      IF (ISET .EQ. 1) VALUE = 1.0
      CONST1 = THICK*0.5D0
      CONST2 = SIP*CONST1
C     WRITE(*,*) 'SIP',SIP, 'CONST2',CONST2,'CONST1',CONST1
      DO 140 K1 = 1, NNEL
          NODE = NOP(K1,NREL)
          X = XYZ(1,NODE)
          Y = XYZ(2,NODE)
          Z = XYZ(3,NODE)
          XXI = XXI + NXI(K1,INTGPN)*(X+VECT(1,3,K1)*CONST2)
          YXI = YXI + NXI(K1,INTGPN)*(Y+VECT(2,3,K1)*CONST2)
          ZXI = ZXI + NXI(K1,INTGPN)*(Z+VECT(3,3,K1)*CONST2)
          XETA = XETA + NETA(K1,INTGPN)*(X+VECT(1,3,K1)*CONST2)
          YETA = YETA + NETA(K1,INTGPN)*(Y+VECT(2,3,K1)*CONST2)
          ZETA = ZETA + NETA(K1,INTGPN)*(Z+VECT(3,3,K1)*CONST2)
          XSI = XSI + N(K1,INTGPN)*VECT(1,3,K1)*CONST1
          YSI = YSI + N(K1,INTGPN)*VECT(2,3,K1)*CONST1
          ZSI = ZSI + N(K1,INTGPN)*VECT(3,3,K1)*CONST1
C
C ----- EVALUATE TWO VECTORS TANGENT TO THE MID SURFACE (SI=0)
C       THESE VACTORS ARE V1={XXI1,YXI1,ZXI1}, V2={XETA1,YETA1,ZETA1}
C
          XXI1 = XXI1 + NXI(K1,INTGPN)*(X+VALUE*CONST2*VECT(1,3,K1))
          YXI1 = YXI1 + NXI(K1,INTGPN)*(Y+VALUE*CONST2*VECT(2,3,K1))
          ZXI1 = ZXI1 + NXI(K1,INTGPN)*(Z+VALUE*CONST2*VECT(3,3,K1))
          XETA1 = XETA1 + NETA(K1,INTGPN)*(X+VALUE*CONST2*VECT(1,3,K1))
          YETA1 = YETA1 + NETA(K1,INTGPN)*(Y+VALUE*CONST2*VECT(2,3,K1))
          ZETA1 = ZETA1 + NETA(K1,INTGPN)*(Z+VALUE*CONST2*VECT(3,3,K1))
  140 CONTINUE
C
C ----- EVALUTE THE CROSS PRODUCT OF V1 AND V2 TO FIND THE NORMAL
C       VECTOR V3
C
      V3(1) = YXI1*ZETA1 - YETA1*ZXI1
      V3(2) = ZXI1*XETA1 - XXI1*ZETA1
      V3(3) = XXI1*YETA1 - YXI1*XETA1
      IF (ISET .EQ. 0) THEN
          VNORM = DSQRT(V3(1)**2+V3(2)**2+V3(3)**2)
          V3(1) = V3(1)/VNORM
          V3(2) = V3(2)/VNORM
          V3(3) = V3(3)/VNORM
      ENDIF
C
      DETJAC = XXI*(YETA*ZSI-ZETA*YSI) - YXI*(XETA*ZSI-ZETA*XSI) + ZXI*(
     1    XETA*YSI-YETA*XSI)
C
C ---- CHECK THE INTEGRITY OF THE TRANSFORMATION FROM ISOP. COORD.
C       TO THE GLOBAL COORDINATES
C
      IF (DETJAC .LE. ZERO) THEN
          WRITE (BUFFER, *) CELEM, NREL
          CALL ERRORS (16, 1, 'JACSHL')
```

```
      ENDIF
C
C*VDIR: ASSUME COUNT(20)
C
      DO 150 K = 1, NNEL
          NX(K) = ((YETA*ZSI-ZETA*YSI)*NXI(K,INTGPN)-(YXI*ZSI-ZXI*YSI)*
     1        NETA(K,INTGPN))/DETJAC
          NY(K) = ((-(XETA*ZSI-ZETA*XSI)*NXI(K,INTGPN))+(XXI*ZSI-ZXI*XSI
     1        )*NETA(K,INTGPN))/DETJAC
          NZ(K) = ((XETA*YSI-YETA*XSI)*NXI(K,INTGPN)-(XXI*YSI-YXI*XSI)*

     1        NETA(K,INTGPN))/DETJAC
  150 CONTINUE
C
      SIX = (YXI*ZETA-ZXI*YETA)/DETJAC
      SIY = -(XXI*ZETA-ZXI*XETA)/DETJAC
      SIZ = (XXI*YETA-YXI*XETA)/DETJAC
      RETURN
      END
C
C ===================== S H N O R M ==============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE SHNORM(ELNUM,NO,NNEL,ITYPE,V1,V2,V3)
C
C ==================================================================
C I
C I   P R O G R A M:
C I
C I   SHNORM EVALUTES THE X_PRIME, Y_PRIME AND Z_PRIME DIRECTIONS
C I   AT THE ELEMENT NODES FOR GENERAL SHELL ELEMENTS
C I
C I
C I   A R G U M E N T    L I S T
C I
C I   ELNUM  = ELEMENT NUMBER
C I   NO     = ELEMENT NODE NUMBER
C I   NNEL   = NUMBER OF NODES IN THE ELEMENT
C I   ITYPE  = ELEMENT TYPE
C I   V1     = VECTOR CONTAINING THE DIRECTION COSINES OF THE X-PRIME
C I   V2     = VECTOR CONTAINING THE DIRECTION COSINES OF THE Y-PRIME
C I   V3     = VECTOR CONTAINING THE DIRECTION COSINES OF THE Z-PRIME
C I
C ==================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C**********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ISOP2D    DIRVEC
C
C**********************************************************
      REAL*4 XYZ,PXI,PETA
      INTEGER  ELNUM
      COMMON/INPUT2/NOP(20,5000)
      COMMON/INPUT3/XYZ(3,10000)
      COMMON/ELLIB2/PXI(9),PETA(9)
      DIMENSION V1( 3 ),V2( 3 ),V3( 3 ),FXI(20),FETA(20)
C
      IT = ITYPE - 100
```

```
      AXI = PXI(NO)
      AETA = PETA(NO)
      CALL ISOP2D (AXI, AETA, FXI, FETA, IT, IERROR)
C
      XXI = 0.0
      XETA = 0.0
      XSI = 0.0
      YXI = 0.0
      YETA = 0.0
      YSI = 0.0
      ZXI = 0.0
      ZETA = 0.0
      ZSI = 0.0
C
C*VDIR: ASSUME COUNT(8)
C
C ----- EVALUATE TWO VECTORS TANGENT TO THE MID SURFACE (SI=0)
C       THESE VACTORS ARE V1={XXI,YXI,ZXI}, V2={XETA,YETA,ZETA}
C
      DO 100 K1 = 1, NNEL
          NODE = NOP(K1,ELNUM)
          X = XYZ(1,NODE)
          Y = XYZ(2,NODE)
          Z = XYZ(3,NODE)
          XXI = XXI + FXI(K1)*X
          YXI = YXI + FXI(K1)*Y
          ZXI = ZXI + FXI(K1)*Z
          XETA = XETA + FETA(K1)*X
          YETA = YETA + FETA(K1)*Y
          ZETA = ZETA + FETA(K1)*Z
  100 CONTINUE
C
C ----- EVALUTE THE CROSS PRODUCT OF V1 AND V2 TO FIND THE NORMAL
C       VECTOR V3
C
      V3(1) = YXI*ZETA - YETA*ZXI
      V3(2) = ZXI*XETA - XXI*ZETA
      V3(3) = XXI*YETA - YXI*XETA
      VNORM = DSQRT(V3(1)**2+V3(2)**2+V3(3)**2)
      V3(1) = V3(1)/VNORM
      V3(2) = V3(2)/VNORM
      V3(3) = V3(3)/VNORM
C
C
C ----- EVALUATE THE COMPLETE ORTHONORMAL COORDINATE SYSTEM.
C       V1 AND V2 EVALUATED NEXT WILL BE DIFFERENT FROM WHAT WAS
C       PRIVIOUSLY DESCRIBED.
C
      CALL DIRVEC (V1, V2, V3)
C
      RETURN
      END
C
C ========================== E L I N F O ============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE ELINFO(ELNUM,ITYPE,NNEL,IFLAG,ISTART,LINES)
C
C ==================================================================
C I
C I    P R O G R A M
C I
C I    PROGRAM 'ELINFO' EXTRACTS ELEMENT INFORMATION FROM THE ARRAY
```

```
C I    'INFOEL'.
C I
C I    A R G U M E N T     L I S T
C I
C I    ELNUM     = ELEMENT NUMBER PASSED BY THE CALLING ROUTINE
C I
C I    ITYPE     = ELEMENT TYPE PASSED TO THE CALLING ROUTINE
C I
C I    NNEL      = NUMBER OF NODES IN THE ELEMNT PASSED TO THE
C I                CALLING ROUTINE
C I
C I    MATNUM    = MATERIAL I.D. NUMBER FOR THE ELEMENT PASSED TO THE
C I                CALLING ROUTINE
C I
C I    ISTART    = STARTING POSITION OF THE LINE CONNECTIVITY DATA
C I                IN ARRAYS 'IS' AND 'IE'.
C I
C I    LINES     = NUMBER OF LINES CONNECTING THE NODES WITHIN THE
C I                ELEMENT
C I
C ==================================================================
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C***********************************************************
      INTEGER ELNUM
      REAL*4 THICK,TH
      REAL*8 LTHICK
      COMMON/LAYERA/ELLYFO(320,5000)
      COMMON/INPUTA/INFOEL(6,5000)
      COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
      DIMENSION THICK(9)
      EQUIVALENCE(ITH,TH)
C     WRITE(*,*) 'ELINFO CALLED'
C
C
C ---- CONDENSED DATA STORAGE IN ARRAY INFOEL.
C
C ---- BIT STORAGE ORGANIZATION FOR INFOEL(1,ELNUM)
C


C       BIT RANGE      MAX. VALUE      DESCRIPTION
C
C       0-2            7               MATERIAL NUMBER
C       3-8            63              NUMBER OF NODES IN THE ELEMENT
C       9-17           511             ELEMENT TYPE NUMBER
C       18-20          7               ANALYSIS TYPE FLAG
C       21-23          7               INTEGRATION POINTS IN XI DIREC.
C       24-26          7               INTEGRATION POINTS IN ETA DIREC.
C       27-30          7               INTEGRATION POINTS IN ZETA DIR.
C
C
C ---- BIT STORAGE ORGANIZATION FOR INFOEL(2,ELNUM)
C
C       BIT RANGE      MAX. VALUE      DESCRIPTION
C
C       0-7            255             INTEGRATION CODE
```

```
C       8-14           255              #. OF LAYERS
C       15-16          3                INTERNAL LAYER FLAG.
C       17-22          63               NUMBER OF LINES CONNECTING NODES
C       23-29          128              STARTING POSITION OF LINE
C                                       CONNECTIVITY IN ARRAYS ISTART &
C                                       IEND
C
C
C ---- TABLE OF INTERNAL ELEMENT TYPE NUMBERS
C
C       TYPE NO.   NO. OF NODES   DESCRIPTION
C
C       204        4              LINEAR ISOPARAMETRIC QUADRILATERAL
C       208        8              QUADRATIC ISOPARAMETRIC QUADRILATERAL
C       209        9              LAGRANGIAN QUADRATIC ISOP. QUAD.
C       219        9              QUADRATIC TO QUBIC ISOP. QUAD.
C       308        8              LINEAR ISOPARAMETRIC BRICK
C       320        20             QUADRATIC ISOPARAMETRIC BRICK
C
      I = INFOEL(1,ELNUM)
      I1 = INFOEL(2,ELNUM)
      LINES = IAND(I1,8257536)/131072
      ISTART = I1/8388608
      IFLAG = IAND(I,1835008)/262144
      ITYPE = IAND(I,261632)/512
      NNEL = IAND(I,508)/8
      RETURN
C
      ENTRY ELINTM(ELNUM,IDENT,INTCOD,NIPXI,NIPETA,NIPSI,MATNUM,THICK)
      I = INFOEL(1,ELNUM)
      I1 = INFOEL(2,ELNUM)
      NNEL = IAND(I,508)/8
      MATNUM = IAND(I,7)
      NIPSI = I/134217728
      NIPETA = IAND(I,117440512)/16777216
      NIPXI = IAND(I,14680064)/2097152
      INTCOD = IAND(I1,255)
      IDENT = INTCOD + NIPXI + NIPETA*10 + NIPSI*100
      NLAYRS = INT(ELLYFO(320,ELNUM))
      IF (IFLAG.EQ.1 .OR. IFLAG.EQ.2 .OR. IFLAG.EQ.4) THEN
          ITH = INFOEL(3,ELNUM)
          THICK(1) = TH
          ITH = INFOEL(4,ELNUM)
          THICK(2) = TH
          ITH = INFOEL(5,ELNUM)
          THICK(3) = TH
          ITH = INFOEL(6,ELNUM)
          THICK(4) = TH
          IF (NNEL .GT. 4) THEN
              THICK(5) = (THICK(1)+THICK(2))*0.50
              THICK(6) = (THICK(2)+THICK(3))*0.50
              THICK(7) = (THICK(3)+THICK(4))*0.50
              THICK(8) = (THICK(4)+THICK(1))*0.50
              THICK(9) = (THICK(1)+THICK(2))*0.25 + (THICK(3)+THICK(4))*
     1            0.25
          ENDIF
      ENDIF
      RETURN
      END
C
C=================================================================
C     INCLUDE(PROCESS)
      SUBROUTINE LYINFO(ILAYER,LTHICK,ZS,MATRL,DCS,NNEL,
```

```
     $                  ELNUM,NIPXI,NIPETA,NIPSI)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C*************************************************************
C
      INTEGER ELNUM
      REAL*8 LTHICK
      COMMON/LAYERA/ELLYFO(320,5000)
      DIMENSION LTHICK(9),ZS(9),DCS(3,3)
C
      INDEX = 21*(ILAYER-1) + 1
      MATRL = ELLYFO(INDEX,ELNUM)
      LTHICK(1) = ELLYFO(INDEX+1,ELNUM)
      LTHICK(2) = ELLYFO(INDEX+2,ELNUM)
      LTHICK(3) = ELLYFO(INDEX+3,ELNUM)
      LTHICK(4) = ELLYFO(INDEX+4,ELNUM)
      IF (NNEL .GT. 4) THEN
          LTHICK(5) = (LTHICK(1)+LTHICK(2))*0.50
          LTHICK(6) = (LTHICK(2)+LTHICK(3))*0.50
          LTHICK(7) = (LTHICK(3)+LTHICK(4))*0.50
          LTHICK(8) = (LTHICK(4)+LTHICK(1))*0.50
          LTHICK(9) = (LTHICK(1)+LTHICK(2))*0.25 + (LTHICK(3)+LTHICK(4))
     1         *0.25
      ENDIF
C
      ZS(1) = ELLYFO(INDEX+5,ELNUM)
      ZS(2) = ELLYFO(INDEX+6,ELNUM)
      ZS(3) = ELLYFO(INDEX+7,ELNUM)
      ZS(4) = ELLYFO(INDEX+8,ELNUM)
      IF (NNEL .GT. 4) THEN
          ZS(5) = (ZS(1)+ZS(2))*0.50
          ZS(6) = (ZS(2)+ZS(3))*0.50
          ZS(7) = (ZS(3)+ZS(4))*0.50
          ZS(8) = (ZS(4)+ZS(1))*0.50
          ZS(9) = (ZS(1)+ZS(2))*0.25 + (ZS(3)+ZS(4))*0.25
      ENDIF
C
      DCS(1,1) = ELLYFO(INDEX+9,ELNUM)
      DCS(1,2) = ELLYFO(INDEX+10,ELNUM)
      DCS(1,3) = ELLYFO(INDEX+11,ELNUM)
      DCS(2,1) = ELLYFO(INDEX+12,ELNUM)
      DCS(2,2) = ELLYFO(INDEX+13,ELNUM)
      DCS(2,3) = ELLYFO(INDEX+14,ELNUM)
      DCS(3,1) = ELLYFO(INDEX+15,ELNUM)
      DCS(3,2) = ELLYFO(INDEX+16,ELNUM)
      DCS(3,3) = ELLYFO(INDEX+17,ELNUM)
C
      NIPXI = INT(ELLYFO(INDEX+18,ELNUM))
      NIPETA = INT(ELLYFO(INDEX+19,ELNUM))
      NIPSI = INT(ELLYFO(INDEX+20,ELNUM))
C
      RETURN
      END
      SUBROUTINE FILES
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
```

```
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C FILEINF
C
C***********************************************************
      CHARACTER*1 SLASH
      CHARACTER*40 SC1,SC2,SC3,SC4,SC5,SC6,SC7,OUT,STAT,PLT,INT,CRK
      CHARACTER*40 IN1,IN2,IN3,STY,ST1,ST2,SPG,SG1,SG2,CK1,AFK
      CHARACTER*40 IN4,IN5,IN6,IN7,IN8,IN9,INA,INB,INC,IND,INE,INF
      CHARACTER*40 SCFP,OFP,OFN,VOLSER,VOL
      LOGICAL*4 THERE
      COMMON/FILE1/KEEP,IDEL,SCFP,OFP,OFN,VOLSER
      COMMON/INPUTG/IFLAG3,IOINTR,IFPLOT
      DATA SLASH/'/'/
C
      ISCFP = 0
      IF (SCFP .EQ. ' ') THEN
          SCFP = 'NLRC3D.SCR'
          ISCFP = 10
      ELSE
          DO 100 K = 1, 40
              IF (SCFP(K:K) .EQ. ' ') THEN
                  ISCFP = K - 1
                  GO TO 110
              ENDIF
  100     CONTINUE
  110     CONTINUE
      ENDIF
C
      SC1 = SLASH//SCFP(1:ISCFP)//'.SC1'
      SC2 = SLASH//SCFP(1:ISCFP)//'.SC2'
      SC3 = SLASH//SCFP(1:ISCFP)//'.SC3'
      SC4 = SLASH//SCFP(1:ISCFP)//'.SC4'
      SC5 = SLASH//SCFP(1:ISCFP)//'.SC5'
      SC6 = SLASH//SCFP(1:ISCFP)//'.SC6'
      SC7 = SLASH//SCFP(1:ISCFP)//'.SC7'
      PLT = SLASH//SCFP(1:ISCFP)//'.PLT'
      INT = SLASH//SCFP(1:ISCFP)//'.INT'
      IN1 = SLASH//SCFP(1:ISCFP)//'.IN1'
      IN2 = SLASH//SCFP(1:ISCFP)//'.IN2'
      IN3 = SLASH//SCFP(1:ISCFP)//'.IN3'
      IN4 = SLASH//SCFP(1:ISCFP)//'.IN4'
      IN5 = SLASH//SCFP(1:ISCFP)//'.IN5'
      IN6 = SLASH//SCFP(1:ISCFP)//'.IN6'
      IN7 = SLASH//SCFP(1:ISCFP)//'.IN7'
      IN8 = SLASH//SCFP(1:ISCFP)//'.IN8'
      IN9 = SLASH//SCFP(1:ISCFP)//'.IN9'
      INA = SLASH//SCFP(1:ISCFP)//'.INA'
      INB = SLASH//SCFP(1:ISCFP)//'.INB'
      INC = SLASH//SCFP(1:ISCFP)//'.INC'
      IND = SLASH//SCFP(1:ISCFP)//'.IND'
      INE = SLASH//SCFP(1:ISCFP)//'.INE'
      INF = SLASH//SCFP(1:ISCFP)//'.INF'
      STY = SLASH//SCFP(1:ISCFP)//'.STY'
      ST1 = SLASH//SCFP(1:ISCFP)//'.ST1'
      ST2 = SLASH//SCFP(1:ISCFP)//'.ST2'
      AFK = SLASH//SCFP(1:ISCFP)//'.AFK'
      SPG = SLASH//SCFP(1:ISCFP)//'.SPG'
      SG1 = SLASH//SCFP(1:ISCFP)//'.SG1'
      SG2 = SLASH//SCFP(1:ISCFP)//'.SG2'
      CK1 = SLASH//SCFP(1:ISCFP)//'.CK1'
```

```
C
      IOFP = 0
      IF (OFN .NE. ' ') THEN
          OUT = SLASH//OFN
          CRK = SLASH//OFN
      ELSE
          DO 120 K = 1, 40
              IF (OFP(K:K) .EQ. ' ') THEN
                  IOFP = K - 1
                  GO TO 130
              ENDIF
  120     CONTINUE
  130     CONTINUE
          OUT = SLASH//OFP(1:IOFP)//'.OUT'
          CRK = SLASH//OFP(1:IOFP)//'.CRK'
      ENDIF
C
      INQUIRE(FILE=SC1, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 240, 'VOLSER', VOL, 'TRK'
     1    , 20, 'SECOND', 40)
      OPEN(1, STATUS='UNKNOWN', FILE=SC1)
C
      INQUIRE(FILE=SC2, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 240, 'VOLSER', VOL, 'TRK'
     1    , 20, 'SECOND', 40)
      OPEN(2, STATUS='UNKNOWN', FILE=SC2)
C
      INQUIRE(FILE=SC3, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 240, 'VOLSER', VOL, 'TRK'
     1    , 20, 'SECOND', 40)
      OPEN(3, STATUS='UNKNOWN', FILE=SC3)
C
      INQUIRE(FILE=SC4, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 240, 'VOLSER', VOL, 'TRK'
     1    , 20, 'SECOND', 40)
      OPEN(4, STATUS='UNKNOWN', FILE=SC4)
C
      INQUIRE(FILE=SC5, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
```

```
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 130, 'VOLSER', VOL, 'TRK'
     1     , 20, 'SECOND', 40)
      OPEN(16, STATUS='UNKNOWN', FILE=SC5)
C
      INQUIRE(FILE=SC6, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 130, 'VOLSER', VOL, 'TRK'
     1     , 20, 'SECOND', 40)
      OPEN(14, STATUS='UNKNOWN', FILE=SC6)
C
      INQUIRE(FILE=SC7, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 130, 'VOLSER', VOL, 'TRK'
     1     , 20, 'SECOND', 40)
      OPEN(15, STATUS='UNKNOWN', FILE=SC7)
C
      INQUIRE(FILE=PLT, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 15, 'VOLSER', VOL, 'TRK'
     1     , 20, 'SECOND', 40)
      OPEN(17, STATUS='UNKNOWN', FILE=PLT)
C
      INQUIRE(FILE=OUT, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 137, 'VOLSER', VOL, 'TRK'
     1     , 20, 'SECOND', 40)
      IF (IFLAG3.EQ.1 .AND. THERE) THEN
          OPEN(13, STATUS='OLD', FILE=OUT)
          ICOUNT = 0
  140     CONTINUE
          READ (13, '(A80)', END=150)
          ICOUNT = ICOUNT + 1
          GO TO 140
  150     CONTINUE
          BACKSPACE 13
          WRITE (*, *) 'IOUT ICOUNT', ICOUNT
      ELSE
          OPEN(13, STATUS='UNKNOWN', FILE=OUT)
      ENDIF
C
      INQUIRE(FILE=INT, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
```

```
     1    , 40, 'SECOND', 50)
      IF (IFLAG3.EQ.1 .AND. THERE) THEN
          OPEN(20, STATUS='OLD', FILE=INT)
  160     CONTINUE
          READ (20, '(A80)', END=170)
          GO TO 160
  170     CONTINUE
          BACKSPACE 20
      ELSE
          OPEN(20, STATUS='UNKNOWN', FILE=INT)
      ENDIF
C
      INQUIRE(FILE=IN1, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
     1    , 40, 'SECOND', 50)
      IF (IFLAG3.EQ.1 .AND. THERE) THEN
          OPEN(21, STATUS='OLD', FILE=IN1)
  180     CONTINUE
          READ (21, '(A80)', END=190)
          GO TO 180
  190     CONTINUE
          BACKSPACE 21
      ELSE
          OPEN(21, STATUS='UNKNOWN', FILE=IN1)
      ENDIF
C
      INQUIRE(FILE=IN2, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
     1    , 40, 'SECOND', 50)
      IF (IFLAG3.EQ.1 .AND. THERE) THEN
          OPEN(22, STATUS='OLD', FILE=IN2)
  200     CONTINUE
          READ (22, '(A80)', END=210)
          GO TO 200
  210     CONTINUE
          BACKSPACE 22
      ELSE
          OPEN(22, STATUS='UNKNOWN', FILE=IN2)
      ENDIF
C
      INQUIRE(FILE=IN3, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
     1    , 40, 'SECOND', 50)
      IF (IFLAG3.EQ.1 .AND. THERE) THEN
          OPEN(23, STATUS='OLD', FILE=IN3)
  220     CONTINUE
          READ (23, '(A80)', END=230)
          GO TO 220
```

```
 230     CONTINUE
         BACKSPACE 23
     ELSE
         OPEN(23, STATUS='UNKNOWN', FILE=IN3)
     ENDIF
C
     INQUIRE(FILE=IN4, EXIST=THERE)

     IF (THERE) THEN
         VOL = ' '
     ELSE
         VOL = VOLSER
     ENDIF
     CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
    1     , 40, 'SECOND', 50)
     IF (IFLAG3.EQ.1 .AND. THERE) THEN
         OPEN(24, STATUS='OLD', FILE=IN4)
 240     CONTINUE
         READ (24, '(A80)', END=250)
         GO TO 240
 250     CONTINUE
         BACKSPACE 24
     ELSE
         OPEN(24, STATUS='UNKNOWN', FILE=IN4)
     ENDIF
C
     INQUIRE(FILE=IN5, EXIST=THERE)
     IF (THERE) THEN
         VOL = ' '
     ELSE
         VOL = VOLSER
     ENDIF
     CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
    1     , 40, 'SECOND', 50)
     IF (IFLAG3.EQ.1 .AND. THERE) THEN
         OPEN(25, STATUS='OLD', FILE=IN5)
 260     CONTINUE
         READ (25, '(A80)', END=270)
         GO TO 260
 270     CONTINUE
         BACKSPACE 25
     ELSE
         OPEN(25, STATUS='UNKNOWN', FILE=IN5)
     ENDIF
C
     INQUIRE(FILE=IN6, EXIST=THERE)
     IF (THERE) THEN
         VOL = ' '
     ELSE
         VOL = VOLSER
     ENDIF
     CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
    1     , 40, 'SECOND', 50)
     IF (IFLAG3.EQ.1 .AND. THERE) THEN
         OPEN(26, STATUS='OLD', FILE=IN6)
 280     CONTINUE
         READ (26, '(A80)', END=290)
         GO TO 280
 290     CONTINUE
         BACKSPACE 26
     ELSE
         OPEN(26, STATUS='UNKNOWN', FILE=IN6)
     ENDIF
```

```
C
      INQUIRE(FILE=IN7, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
     1     , 40, 'SECOND', 50)
      IF (IFLAG3.EQ.1 .AND. THERE) THEN
          OPEN(27, STATUS='OLD', FILE=IN7)
  300     CONTINUE
          READ (27, '(A80)', END=310)
          GO TO 300
  310     CONTINUE
          BACKSPACE 27
      ELSE
          OPEN(27, STATUS='UNKNOWN', FILE=IN7)
      ENDIF
C
      INQUIRE(FILE=IN8, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
     1     , 40, 'SECOND', 50)
      IF (IFLAG3.EQ.1 .AND. THERE) THEN
          OPEN(28, STATUS='OLD', FILE=IN8)
  320     CONTINUE
          READ (28, '(A80)', END=330)
          GO TO 320
  330     CONTINUE
          BACKSPACE 28
      ELSE
          OPEN(28, STATUS='UNKNOWN', FILE=IN8)
      ENDIF
C
      INQUIRE(FILE=IN9, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
     1     , 40, 'SECOND', 50)
      IF (IFLAG3.EQ.1 .AND. THERE) THEN
          OPEN(29, STATUS='OLD', FILE=IN9)
  340     CONTINUE
          READ (29, '(A80)', END=350)
          GO TO 340
  350     CONTINUE
          BACKSPACE 29
      ELSE
          OPEN(29, STATUS='UNKNOWN', FILE=IN9)
      ENDIF
C
      INQUIRE(FILE=INA, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
```

```
      ENDIF
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
     1    , 40, 'SECOND', 50)
      IF (IFLAG3.EQ.1 .AND. THERE) THEN
          OPEN(30, STATUS='OLD', FILE=INA)
 360      CONTINUE
          READ (30, '(A80)', END=370)
          GO TO 360
 370      CONTINUE
          BACKSPACE 30
      ELSE
          OPEN(30, STATUS='UNKNOWN', FILE=INA)
      ENDIF
C
      INQUIRE(FILE=INB, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
     1    , 40, 'SECOND', 50)
      IF (IFLAG3.EQ.1 .AND. THERE) THEN
          OPEN(31, STATUS='OLD', FILE=INB)
 380      CONTINUE
          READ (31, '(A80)', END=390)
          GO TO 380
 390      CONTINUE
          BACKSPACE 31
      ELSE
          OPEN(31, STATUS='UNKNOWN', FILE=INB)
      ENDIF
C
      INQUIRE(FILE=INC, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
     1    , 40, 'SECOND', 50)
      IF (IFLAG3.EQ.1 .AND. THERE) THEN
          OPEN(32, STATUS='OLD', FILE=INC)
 400      CONTINUE
          READ (32, '(A80)', END=410)
          GO TO 400
 410      CONTINUE
          BACKSPACE 32
      ELSE
          OPEN(32, STATUS='UNKNOWN', FILE=INC)
      ENDIF
C
      INQUIRE(FILE=IND, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
     1    , 40, 'SECOND', 50)
      IF (IFLAG3.EQ.1 .AND. THERE) THEN
          OPEN(33, STATUS='OLD', FILE=IND)
 420      CONTINUE
```

```
          READ (33, '(A80)', END=430)
          GO TO 420
  430     CONTINUE
          BACKSPACE 33
      ELSE
          OPEN(33, STATUS='UNKNOWN', FILE=IND)
      ENDIF
C
      INQUIRE(FILE=INE, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
     1     , 40, 'SECOND', 50)
      IF (IFLAG3.EQ.1 .AND. THERE) THEN
          OPEN(34, STATUS='OLD', FILE=INE)
  440     CONTINUE
          READ (34, '(A80)', END=450)
          GO TO 440
  450     CONTINUE
          BACKSPACE 34
      ELSE
          OPEN(34, STATUS='UNKNOWN', FILE=INE)
      ENDIF
C
      INQUIRE(FILE=INF, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
     1     , 40, 'SECOND', 50)
      IF (IFLAG3.EQ.1 .AND. THERE) THEN
          OPEN(35, STATUS='OLD', FILE=INF)
  460     CONTINUE
          READ (35, '(A80)', END=470)
          GO TO 460
  470     CONTINUE
          BACKSPACE 35
      ELSE
          OPEN(35, STATUS='UNKNOWN', FILE=INF)
      ENDIF
C
      INQUIRE(FILE=STY, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'VS', 'LRECL', -1, 'VOLSER', VOL, 'TRK'
     1     , 100, 'SECOND', 100, 'BLKSIZE', 32760)
      OPEN(50, STATUS='UNKNOWN', FILE=STY)
C
      INQUIRE(FILE=ST1, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'VS', 'LRECL', -1, 'VOLSER', VOL, 'TRK'
```

```
     1    , 100, 'SECOND', 100, 'BLKSIZE', 32760)
      OPEN(60, STATUS='UNKNOWN', FILE=ST1)
C
      INQUIRE(FILE=ST2, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'VS', 'LRECL', -1, 'VOLSER', VOL, 'TRK'
     1    , 100, 'SECOND', 100, 'BLKSIZE', 32760)
      OPEN(61, STATUS='UNKNOWN', FILE=ST2)
C
      INQUIRE(FILE=AFK, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'VS', 'LRECL', -1, 'VOLSER', VOL, 'TRK'
     1    , 100, 'SECOND', 100, 'BLKSIZE', 32760)
      OPEN(62, STATUS='UNKNOWN', FILE=AFK)
C
      INQUIRE(FILE=SPG, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'VS', 'LRECL', -1, 'VOLSER', VOL, 'TRK'
     1    , 100, 'SECOND', 100, 'BLKSIZE', 32760)
      OPEN(80, STATUS='UNKNOWN', FILE=SPG)
C
      INQUIRE(FILE=SG1, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'VS', 'LRECL', -1, 'VOLSER', VOL, 'TRK'
     1    , 100, 'SECOND', 100, 'BLKSIZE', 32760)
      OPEN(70, STATUS='UNKNOWN', FILE=SG1)
C
      INQUIRE(FILE=SG2, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'VS', 'LRECL', -1, 'VOLSER', VOL, 'TRK'
     1    , 100, 'SECOND', 100, 'BLKSIZE', 32760)
      OPEN(71, STATUS='UNKNOWN', FILE=SG2)
C
      INQUIRE(FILE=CK1, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
     1    , 40, 'SECOND', 50)
      IF (IFLAG3.EQ.1 .AND. THERE) THEN
          OPEN(72, STATUS='OLD', FILE=CK1)
```

```
 480      CONTINUE
          READ (72, '(A80)', END=490)
          GO TO 480
 490      CONTINUE
          BACKSPACE 72
      ELSE
          OPEN(72, STATUS='UNKNOWN', FILE=CK1)
      ENDIF
C
      INQUIRE(FILE=CRK, EXIST=THERE)
      IF (THERE) THEN
          VOL = ' '
      ELSE
          VOL = VOLSER
      ENDIF
      CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 137, 'VOLSER', VOL, 'TRK'
     1    , 100, 'SECOND', 100)
      IF (IFLAG3.EQ.1 .AND. THERE) THEN
          OPEN(40, STATUS='OLD', FILE=CRK)
 500      CONTINUE
          READ (40, '(A80)', END=510)
          GO TO 500
 510      CONTINUE
          BACKSPACE 40
      ELSE
          OPEN(40, STATUS='UNKNOWN', FILE=CRK)
      ENDIF
C
C     WRITE(*,*) 'EXITING FILES'
      RETURN
C
C
C
      ENTRY CFILE
C
      IF (KEEP.EQ.0 .OR. IDEL.EQ.1) THEN
          STAT = 'DELETE'
      ELSE
          STAT = 'KEEP'
      ENDIF
C
      CLOSE(1, STATUS=STAT)
      CLOSE(2, STATUS=STAT)
      CLOSE(3, STATUS=STAT)
      CLOSE(4, STATUS=STAT)
      CLOSE(16, STATUS=STAT)
      CLOSE(14, STATUS=STAT)
      CLOSE(15, STATUS=STAT)
      CLOSE(50, STATUS=STAT)
      CLOSE(60, STATUS=STAT)
      CLOSE(61, STATUS=STAT)
      CLOSE(62, STATUS=STAT)
      CLOSE(80, STATUS=STAT)
      CLOSE(70, STATUS=STAT)
      CLOSE(71, STATUS=STAT)
      CLOSE(17, STATUS='KEEP')
      CLOSE(13, STATUS='KEEP')
      CLOSE(40, STATUS='KEEP')
      CLOSE(72, STATUS='KEEP')
      CLOSE(18, STATUS='KEEP')
      INQUIRE(FILE=INT, EXIST=THERE)
      IF (THERE) THEN
          CLOSE(20, STATUS='KEEP')
```

```
      ELSE
          CLOSE(20, STATUS='DELETE')
      ENDIF
C
      INQUIRE(FILE=IN1, EXIST=THERE)
      IF (THERE) THEN
          CLOSE(21, STATUS='KEEP')
      ELSE
          CLOSE(21, STATUS='DELETE')
      ENDIF
C
      INQUIRE(FILE=IN2, EXIST=THERE)
      IF (THERE) THEN
          CLOSE(22, STATUS='KEEP')
      ELSE
          CLOSE(22, STATUS='DELETE')
      ENDIF
C
      INQUIRE(FILE=IN3, EXIST=THERE)
      IF (THERE) THEN
          CLOSE(23, STATUS='KEEP')
      ELSE
          CLOSE(23, STATUS='DELETE')
      ENDIF
C
      INQUIRE(FILE=IN4, EXIST=THERE)
      IF (THERE) THEN
          CLOSE(24, STATUS='KEEP')
      ELSE
          CLOSE(24, STATUS='DELETE')


      ENDIF
C
      INQUIRE(FILE=IN5, EXIST=THERE)
      IF (THERE) THEN
          CLOSE(25, STATUS='KEEP')
      ELSE
          CLOSE(25, STATUS='DELETE')
      ENDIF
C
      INQUIRE(FILE=IN6, EXIST=THERE)
      IF (THERE) THEN
          CLOSE(26, STATUS='KEEP')
      ELSE
          CLOSE(26, STATUS='DELETE')
      ENDIF
C
      INQUIRE(FILE=IN7, EXIST=THERE)
      IF (THERE) THEN
          CLOSE(27, STATUS='KEEP')
      ELSE
          CLOSE(27, STATUS='DELETE')
      ENDIF
C
      INQUIRE(FILE=IN8, EXIST=THERE)
      IF (THERE) THEN
          CLOSE(28, STATUS='KEEP')
      ELSE
          CLOSE(28, STATUS='DELETE')
      ENDIF
C
      INQUIRE(FILE=IN9, EXIST=THERE)
      IF (THERE) THEN
```

```
        CLOSE(29, STATUS='KEEP')
      ELSE
        CLOSE(29, STATUS='DELETE')
      ENDIF
C
      INQUIRE(FILE=INA, EXIST=THERE)
      IF (THERE) THEN
        CLOSE(30, STATUS='KEEP')
      ELSE
        CLOSE(30, STATUS='DELETE')
      ENDIF
C
      INQUIRE(FILE=INB, EXIST=THERE)
      IF (THERE) THEN
        CLOSE(31, STATUS='KEEP')
      ELSE
        CLOSE(31, STATUS='DELETE')
      ENDIF
C
      INQUIRE(FILE=INC, EXIST=THERE)
      IF (THERE) THEN
        CLOSE(32, STATUS='KEEP')
      ELSE
        CLOSE(32, STATUS='DELETE')
      ENDIF
C
      INQUIRE(FILE=IND, EXIST=THERE)
      IF (THERE) THEN
        CLOSE(33, STATUS='KEEP')
      ELSE
        CLOSE(33, STATUS='DELETE')
      ENDIF
C
      INQUIRE(FILE=INE, EXIST=THERE)
      IF (THERE) THEN
        CLOSE(34, STATUS='KEEP')
      ELSE
        CLOSE(34, STATUS='DELETE')
      ENDIF
C
      INQUIRE(FILE=INF, EXIST=THERE)
      IF (THERE) THEN
        CLOSE(35, STATUS='KEEP')
      ELSE
        CLOSE(35, STATUS='DELETE')
      ENDIF
C
      RETURN
C
      ENTRY ERRFIL
      OPEN(18, ACCESS='DIRECT', RECL=80, FILE='/CERAMA.NLRC3D.ERR')
      RETURN
C
      END
C     INCLUDE (PROCESS)
      SUBROUTINE INITAL
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
```

```
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C*************************************************************
      REAL*4 XII,ETAI,SII,FMAG,DMAG,SHELLZ
      CHARACTER*80 GTITLE
      COMMON/MAIN1/U(60000),RE1(60000)
      COMMON/MAIN4/RE(60000)
      COMMON/INPUT7/RX(60000),RY(60000),RZ(60000)
      COMMON/INPUTE/ISPB(10000)
      COMMON/IREP1/IREP(12000),LREP(12000)
      COMMON/INPUTA/INFOEL(6,5000)
      COMMON/SKTR2/SHELLZ(3 , 10000)
      COMMON/SAFE2/R(60000),IDOF(60000),JDIAG(60000)
      COMMON/SAFE4/IRDOF(60000),IUDOF(60000)
      COMMON/PATH/HISTX(10,30),HISTY(10,30),IPATH(10)
      COMMON/MPCS/COEFMP(40000),ALAMB(40000),MPCDOF(40000),
     $           MPCADR(2,5000),NMPC,MPCPNT,MAXMPC
      COMMON/OUTPT2/IOEL(5000),IONOD(10000)
      COMMON/ICKPIF/ICKFGO,ICKNTR
      ICKFGO = 0
      ICKNTR = 1
C
C
      MAXMPC = 5000
C
      DO IJ = 1, 10
          IPATH(10) = 0
          DO KL = 1, 30
              HISTX(IJ,KL) = 0.0
              HISTY(IJ,KL) = 0.0
          END DO
      END DO
      IPATH(1) = 2
      IPATH(2) = 2
      IPATH(3) = 2
      HISTX(2,1) = 1.0
      HISTY(2,1) = 1.0
      HISTX(2,2) = 1.0
      HISTY(2,2) = 1.0
      HISTX(3,1) = 0.0
      HISTY(3,1) = 0.0
      HISTX(3,2) = 1.0
      HISTY(3,2) = 1.0
C
      DO KP = 1, 40000
          MPCDOF(KP) = 0
          COEFMP(KP) = 0.0
          ALAMB(KP) = 0.0
      END DO
      DO KB = 1, 5000
          MPCADR(1,KB) = 0
          MPCADR(2,KB) = 0
      END DO
      DO K1 = 1, 10000
          IONOD(K1) = 0
          ISPB(K1) = 0
          SHELLZ(1,K1) = 0.
          SHELLZ(2,K1) = 0.
          SHELLZ(3,K1) = 0.
      END DO
C
      DO K1 = 1, 5000
          IOEL(K1) = 0
```

```
            DO K2 = 1, 6
               INFOEL(K2,K1) = 0.
            END DO
         END DO
C
         DO K1 = 1, 60000
            IDOF(K1) = 0
            JDIAG(K1) = 0
            R(K1) = 0.
            U(K1) = 0.
            IRDOF(K1) = 0.
            IUDOF(K1) = 0.
            RX(K1) = 0.
            RY(K1) = 0.
            RZ(K1) = 0.
            RE(K1) = 0.
            RE1(K1) = 0.
         END DO
C
         DO K1 = 1, 12000
            IREP(K1) = 0
            LREP(K1) = 0
         END DO
         RETURN
         END
C
C ======================= I N P U T ================================
C
C        INCLUDE (PROCESS)
         SUBROUTINE INPUT(IOUT,IERROR)
C
C =================================================================
C I
C I      P R O G R A M:
C I
C I      INPUT is the main input routine for NLRC3D. This program reads
C I      each record of the input file and extracts the command by
C I      calling subroutine COMPRO. The extracted commands are then
C I      processed appropriately.
C I
C I
C I      O N   E N T R Y:
C I
C I      IOUT   = output device number
C I
C I
C I      O N   R E T U R N:
C I
C I      IDOF   = is the array containing the degree of freedom
C I               information for each node.
C I               =1; degree of freedom is constrained (zero displ.)
C I               =0; degree of freedom is unconstrained
C I               =-1; degree of freedom has a prescribed value and
C I                    is constrained (non-zero displacement)
C I
C =================================================================
C
         IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C**************************************************************
C
```

```
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C EBORAS    COMPRO    IODISP    IOCNST    CNPROP    IOLOAD
C SULOAD    GRAPHX    IOMATR    INANDY    NLPROP    IOHIST
C ELEMEN    IOFREE    IOOUT     FILES     CAEDS     ERRORS
C OUT11
C
C**************************************************************
      REAL*8 LX,LY
      REAL*4 XYZ,THICK
      CHARACTER*80 BUFFER,BUFF,TITLE
      CHARACTER*40 SCFP,OFP,OFN,VOLSER,UNIVER
      CHARACTER*6  COMM
      COMMON/INPUT9/THICK(9),IFLAG
      COMMON/MAIN1/U(60000),RE1(60000)
      COMMON/INPUT2/NOP(20,5000)
      COMMON/INPUT3/XYZ(3,10000)
C
C
C ---- Common INPUT7 is used with a different organization in module
C       SAFE. IT IS USED HERE FOR THE PURPOSE OF SAVING STORAGE FOR
C       holding some temperary parameters. Its size should be
C       consistent with its specification in module safe
C
      COMMON/INPUT7/NNELEM(360000)
C
C ---- nnelem  = number of nodes for the element
C
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1              NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC1(10),LDCONT,INCPTR
      COMMON/INPUTB/FAC,FACNEW,FACLOW,FACHIG,ENRG1,NDIVER,ISTOP
      COMMON/INPUTC/TITLE
      COMMON/INPUTE/ISPB(10000)
      COMMON/INPUTG/IFLAG3,IOINTR,IFPLOT
      COMMON/BOUND1/NCURVS
      COMMON/BOUND2/XZL(6),XZR(6),YZB(6),YZT(6)
      COMMON/INPUTI/INTFAC(500)
      COMMON/FILE1/KEEP,IDEL,SCFP,OFP,OFN,VOLSER
      COMMON/COMP2/COMM,BUFFER,BUFF
      COMMON/POINTS/LX(4 , 6) , LY(4 , 6)
      COMMON/SAFE2/R(60000),IDOF(60000),JDIAG(60000)
      COMMON/MPCS/COEFMP(40000),ALAMB(40000),MPCDOF(40000),
     $           MPCADR(2,5000),NMPC,MPCPNT,MAXMPC
      COMMON/LPROP/PROPER(25,15)
      COMMON/ICKIPF/ICKFGO,ICKNTR
      COMMON/ICONSA/IFLAG7
      DIMENSION DUMMY(6),M(20),LAMINA(5000,2)
      CALL EBORAS
C
C ---- read one line of the input file and store in BUFFER
C
C
C .... initialize the output vector for elements/nodes
c
      LDEV11 = 11
      NCURVE = 0
      LDCONT = 0
      K1 = 0
      MAXMPC = 5000
      IFLAG7 = 0
      NMPC = 0
      MPCPNT = 0
```

```
      INCPTR = 1
      IFLAG3 = 0
      DO IK = 1, 5000
         LAMINA(IK,1) = 0.0
         LAMINA(IK,2) = 0.0
      END DO
  110 CONTINUE
      READ (LDEV11, '(A80)', END=300) BUFFER
C
C ----  extract the first six characters of each command in the BUFFER
C        and place all variables associated with each command in the
C        internal file BUFF.
C
  120 CONTINUE
      CALL COMPRO (N, NVAR)
C
  130 CONTINUE
      IF (N .EQ. 0) THEN
         ASSIGN 110 TO NEXT
      ELSE
         ASSIGN 120 TO NEXT
      ENDIF
C
      IF (COMM .EQ. 'TITLE') THEN
         GO TO 150
      ELSE IF (COMM .EQ. 'COMMEN') THEN          ! COMMENT LINE
         GO TO 110
      ELSE IF (COMM.EQ.'COORDI' .OR. COMM.EQ.'NODES') THEN
         GO TO 170
      ELSE IF (COMM.EQ.'NODE' .OR. COMM.EQ.'JOINTS') THEN
         GO TO 170
      ELSE IF (COMM.EQ.'MEMBER' .OR. COMM.EQ.'INCIDE') THEN
         GO TO 210
      ELSE IF (COMM .EQ. 'CONNEC') THEN
         GO TO 210
      ELSE IF (COMM .EQ. 'DISPLA') THEN
         CALL IODISP (IDOF, NNDF, IDIM, N, ICOMM, LDEV11, IOUT)
         IF (ICOMM .EQ. 1) THEN
            GO TO 130
         ELSE IF (ICOMM .EQ. 2) THEN
            GO TO 300
         ELSE
            GO TO 110
         ENDIF
      ELSE IF (COMM .EQ. 'CONSTR') THEN
         CALL IOCNST (LAMINA, IOUT, IERROR, ICOMM)
         IF (IFLAG3 .EQ. 0) CALL CNPROP (LAMINA, IOUT, IERROR)
         IF (ICOMM .EQ. 1) THEN
            GO TO 130
         ELSE IF (ICOMM .EQ. 2) THEN
            GO TO 300
         ELSE
            GO TO 110
         ENDIF
      ELSE IF (COMM.EQ.'LOAD' .OR. COMM.EQ.'LOADS') THEN
         CALL IOLOAD (IDIM, N, NNDF, ICOMM, LDEV11, IOUT)
         IF (ICOMM .EQ. 1) THEN
            GO TO 130
         ELSE IF (ICOMM .EQ. 2) THEN
            GO TO 300
         ELSE
            GO TO 110
         ENDIF
```

```
ELSE IF (COMM.EQ.'SURFAC' .OR. COMM.EQ.'SU_LOD') THEN
    CALL SULOAD (IDIM, N, NNDF, ICOMM, LDEV11, IOUT)
    IF (ICOMM .EQ. 1) THEN
        GO TO 130
    ELSE IF (ICOMM .EQ. 2) THEN
        GO TO 300
    ELSE
        GO TO 110
    ENDIF
ELSE IF (COMM .EQ. 'GRAPHI') THEN
    READ (BUFF, *, END=290) IFPLOT
    CALL GRAPHX (N, ICOMM, LDEV11, IOUT)
    IF (ICOMM .EQ. 1) THEN
        GO TO 130
    ELSE IF (ICOMM .EQ. 2) THEN
        GO TO 300
    ELSE
        GO TO 110
    ENDIF
ENDIF
IF (COMM.EQ.'MATERI' .OR. COMM.EQ.'MAT') THEN
    CALL IOMATR (N, ICOMM, LDEV11, IOUT)
    IF (ICOMM .EQ. 1) THEN
        GO TO 130
    ELSE IF (ICOMM .EQ. 2) THEN
        GO TO 300
    ELSE
        GO TO 110
    ENDIF
ELSE IF (COMM .EQ. 'STOP') THEN
    READ (BUFF, *, END=290) ISTOP
ELSE IF (COMM .EQ. 'LINEAR') THEN
    IFLAG1 = 0
ELSE IF (COMM .EQ. 'NONLIN') THEN
    IFLAG1 = 1
ELSE IF (COMM .EQ. 'NONSYM') THEN
    IFLAG2 = 1
ELSE IF (COMM .EQ. 'SYMMET') THEN
    IFLAG2 = 0
ELSE IF (COMM .EQ. 'RCPREP') THEN
    CALL INANDY (PROPER, IOUT, IERROR, MATNUM)
    IF (IFLAG3 .EQ. 0) CALL NLPROP (IOUT, IERROR, MATNUM)
ELSE IF (COMM.EQ.'HISTOR' .OR. COMM.EQ.'HIST') THEN
    CALL IOHIST (IOUT, IERROR)
ELSE IF (COMM.EQ.'ELEMEN' .OR. COMM.EQ.'ELEM') THEN
    CALL ELEMEN (N, ICOMM, LDEV11, IOUT, NNELEM)
    IF (ICOMM .EQ. 1) THEN
        GO TO 130
    ELSE IF (ICOMM .EQ. 2) THEN
        GO TO 300
    ELSE
        GO TO 110
    ENDIF


ELSE IF (COMM .EQ. 'FREEBO') THEN
    CALL IOFREE (N, ICOMM, LDEV11, IOUT)
    IF (ICOMM .EQ. 1) THEN
        GO TO 130
    ELSE IF (ICOMM .EQ. 2) THEN
        GO TO 300
    ELSE
        GO TO 110
    ENDIF
```

```
      ELSE IF (COMM .EQ. 'DIMENS') THEN
          READ (BUFF, *, END=290) IDIM
          IF (IDIM .EQ. 3) THEN
              NNDF = 6
          ELSE
              NNDF = IDIM
          ENDIF
      ELSE IF (COMM .EQ. 'ITERAT') THEN
          READ (BUFF, *, END=290) MNIT
      ELSE IF (COMM .EQ. 'INCREM') THEN
          LDCONT = LDCONT + 1
          READ (BUFF, *, END=290) NLINC1(LDCONT)
      ELSE IF (COMM .EQ. 'INCRE1') THEN
          READ (BUFF, *, END=290) NLINC
      ELSE IF (COMM .EQ. 'ICKNTR') THEN
          READ (BUFF, *, END=290) ICKNTR
      ELSE IF (COMM.EQ.'FRACTI' .OR. COMM.EQ.'FRACT') THEN
          READ (BUFF, *, END=290) FRACT(LDCONT)
      ELSE IF (COMM .EQ. 'INCPTR') THEN
          READ (BUFF, *, END=290) INCPTR
      ELSE IF (COMM .EQ. 'FACLOW') THEN
          READ (BUFF, *, END=290) FACLOW
          FAC = FACLOW
      ELSE IF (COMM .EQ. 'FACHIG') THEN
          READ (BUFF, *, END=290) FACHIG
      ELSE IF (COMM .EQ. 'RESTAR') THEN
          IFLAG3 = 1
      ELSE IF (COMM .EQ. 'EXIT') THEN
          IFLAG7 = 1
      ELSE IF (COMM .EQ. 'OUTPUT') THEN
          CALL IOOUT (N, ICOMM, LDEV11)
          IF (ICOMM .EQ. 1) THEN
              GO TO 130
          ELSE IF (ICOMM .EQ. 2) THEN
              GO TO 300
          ELSE
              GO TO 110
          ENDIF
      ELSE IF (COMM .EQ. 'INTERF') THEN
          GO TO 260
      ELSE
          GO TO 140
      ENDIF
      GO TO NEXT
  140 CONTINUE
      IF (COMM .EQ. 'PAX') THEN
          READ (BUFF, *, END=290) LX(1,NCURVE)
      ELSE IF (COMM .EQ. 'PAY') THEN
          READ (BUFF, *, END=290) LY(1,NCURVE)
      ELSE IF (COMM .EQ. 'PBX') THEN
          READ (BUFF, *, END=290) LX(2,NCURVE)
      ELSE IF (COMM .EQ. 'PBY') THEN
          READ (BUFF, *, END=290) LY(2,NCURVE)
      ELSE IF (COMM .EQ. 'RAX') THEN
          READ (BUFF, *, END=290) LX(3,NCURVE)
      ELSE IF (COMM .EQ. 'RAY') THEN
          READ (BUFF, *, END=290) LY(3,NCURVE)
      ELSE IF (COMM .EQ. 'RBX') THEN
          READ (BUFF, *, END=290) LX(4,NCURVE)
      ELSE IF (COMM .EQ. 'RBY') THEN
          READ (BUFF, *, END=290) LY(4,NCURVE)
      ELSE IF (COMM .EQ. 'CURVE') THEN
          READ (BUFF, *, END=290) NCURVE
```

```
          NCURVS = MAX0(NCURVE,NCURVS)
      ELSE IF (COMM .EQ. 'XZL') THEN
          READ (BUFF, *, END=290) XZL(NCURVE)
      ELSE IF (COMM .EQ. 'XZR') THEN
          READ (BUFF, *, END=290) XZR(NCURVE)
      ELSE IF (COMM .EQ. 'YZB') THEN
          READ (BUFF, *, END=290) YZB(NCURVE)
      ELSE IF (COMM .EQ. 'YZT') THEN
          READ (BUFF, *, END=290) YZT(NCURVE)
      ELSE IF (COMM .EQ. 'SCFP') THEN
          READ (BUFF, *, END=290) SCFP
      ELSE IF (COMM .EQ. 'OFP') THEN
          READ (BUFF, *, END=290) OFP
      ELSE IF (COMM .EQ. 'OFN') THEN
          READ (BUFF, *, END=290) OFN
      ELSE IF (COMM .EQ. 'VOLSER') THEN
          READ (BUFF, *, END=290) VOLSER
      ELSE IF (COMM .EQ. 'KEEP') THEN
          KEEP = 1
      ELSE IF (COMM.EQ.'DEL' .OR. COMM.EQ.'DELETE') THEN
          IDEL = 1
      ELSE IF (COMM .EQ. 'BEGIN') THEN
          CALL FILES
      ELSE IF (COMM .EQ. 'CAEDS') THEN
          READ (BUFF, *, END=290) UNIVER
c changes 7/18
          CALL CAEDS (IDOF, U, R, IOUT, UNIVER)
      ELSE
          CALL ERRORS (2, 1, 'INPUT ')
      ENDIF
      GO TO NEXT
C
C ---- read the title cards from input file and echo into output files
C
  150 CONTINUE
      READ (BUFF, *, END=290) NUMBER
      DO 160 K = 1, NUMBER
          READ (LDEV11, '(A80)') TITLE
          WRITE (IOUT, '(A80)') TITLE
  160 CONTINUE
      GO TO NEXT
C
C ---- read and generate the nodal coordinates
C
  170 CONTINUE
      I = 0
      READ (BUFF, *, END=290) NNODES
  180 CONTINUE
      READ (LDEV11, *) K, (DUMMY(IDIR), IDIR = 1, IDIM), INCR
C
C ---- Set bit 10 of ISPB to 1 to indicate that the position of the
C       node has been defined.
C
      ISPB(K) = IBSET(ISPB(K),10)
      XYZ(1,K) = DUMMY(1)
      XYZ(2,K) = DUMMY(2)
      XYZ(3,K) = DUMMY(3)
C
      I = I + 1
      IF (INCR .EQ. 0) GO TO 200
      N = (K-K1)/INCR
      DX = (XYZ(1,K)-XYZ(1,K1))/N
      DY = (XYZ(2,K)-XYZ(2,K1))/N
```

```
      DZ = (XYZ(3,K)-XYZ(3,K1))/N
      K2 = K - INCR
      DO 190 J = K1, K2, INCR
          ISPB(J) = IBSET(ISPB(J),10)
          N1 = (J-K1)/INCR
          XYZ(1,J) = XYZ(1,K1) + N1*DX
          XYZ(2,J) = XYZ(2,K1) + N1*DY
          XYZ(3,J) = XYZ(3,K1) + N1*DZ
          I = I + 1
  190 CONTINUE
      I = I - 1
  200 CONTINUE
      K1 = K
      IF (I .LT. NNODES) GO TO 180
C
      GO TO NEXT
C
C ---- read and generate the element connectivity
C
  210 CONTINUE
      I = 0
      READ (BUFF, *, END=290) NELEM
  220 CONTINUE
      READ (LDEV11, '(A80)', END=300) BUFFER
      READ (BUFFER, *) K
      NNEL = NNELEM(K)
      IF (NNEL .LE. 0) CALL ERRORS (11, 1, 'INPUT ')
      READ (BUFFER, *) K, (NOP(NODE,K), NODE = 1, NNEL), INCR
      I = I + 1
C
C ---- Set bit 20 of ISPB to 1 to indicate that the connectivity for
C       the node has been defined.
C
      ISPB(K) = IBSET(ISPB(K),20)
      IF (INCR .EQ. 0) THEN
          K1 = K
      ELSE
          K2 = (K-K1)/INCR
          DO 230 NODE = 1, NNEL
              M(NODE) = (NOP(NODE,K)-NOP(NODE,K1))/K2
  230     CONTINUE
C
          DO 250 IELEM = K1 + INCR, K - INCR, INCR
              I = I + 1
              IELEM1 = IELEM - INCR
              ISPB(IELEM) = IBSET(ISPB(IELEM),20)
              DO 240 NODE = 1, NNEL
                  NOP(NODE,IELEM) = NOP(NODE,IELEM1) + M(NODE)
  240         CONTINUE
  250     CONTINUE
      ENDIF
      IF (I .LT. NELEM) GO TO 220
      GO TO NEXT
C
C ---- read and generate the interface nodes
C
  260 CONTINUE
      I = 0
      READ (BUFF, *, END=290) NINODE
  270 CONTINUE
      READ (LDEV11, *) K, INCR
      IF (INCR .EQ. 0) THEN
          I = I + 1
```

```
            INTFAC(I) = K
        ELSE
            ISTART = INTFAC(I) + INCR
            IEND = K
            DO 280 J = ISTART, IEND, INCR
                I = I + 1
                INTFAC(I) = J
  280       CONTINUE
        ENDIF
        IF (I .LT. NINODE) GO TO 270
        GO TO NEXT
C
  290 CONTINUE
      CALL ERRORS (3, 1, 'INPUT ')
  300 CONTINUE
      CALL OUT11 (IOUT)
      RETURN
      END
C
C ====================== I O D I S P ============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE IODISP(IDOF,NNDF,IDIM,N,ICOMM,LDEV11,IOUT)
C
C ==================================================================
C I
C I   P R O G R A M:
C I
C I   IODISP reads and stores the nodal displacements in either the
C I   global or the local coordinate system.
C I
C I
C I   O N   E N T R Y:
C I
C I   NNDF   = number of nodal degrees of freedom
C I   IDIM   = physical dimension of the problem (i.e. 2D or 3D)
C I   N      = 0; buffer contains no additional commands
C I            1; Buffer containes at least one additional command
C I   LDEV11 = unit number for the input file
C I   IOUT   = unit number for the output file
C I
C I
C I   O N   R E T U R N:
C I
C I   IDOF   = array containing the nodal restraint information for
C I            each degree of freedom.
C I            =1; D.O.F.  restrained with zero displacement or rot.
C I            =0; D.O.F.  is free
C I            =-1; D.O.F. is restrained with non zero displacements
C I
C I   N      = 0; buffer contains no additional commands
C I            1; Buffer containes at least one additional command
C I
C I   ICOMM  = identifies the action to be taken when control is
C I            returned to the calling routine .
C I            =0; read the next input line
C I            =1; The current command was not resolved by IODIS.
C I                Calling routine should process this command
C I            =2; end of file is reached, do not try to read any
C I                more lines.
C I
C =================================================================
C
```

```
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C COMPRO    IDCORD    ERRORS
C
C************************************************************
      LOGICAL*4 ISP
      CHARACTER*80 BUFFER,BUFF
      CHARACTER*6 COMM
      REAL*4 XAX,YAX,ZAX,XAXIS,YAXIS
      REAL*4 XAXR,YAXR,ZAXR
      COMMON/MAIN1/U(60000),RE1(60000)
      COMMON/INPUTD/XAXIS(4,10000),YAXIS(4,10000)
      COMMON/INPUTE/ISPB(10000)
      COMMON/COMP2/COMM,BUFFER,BUFF
      COMMON/SAFE4/IRDOF(60000),IUDOF(60000)
      DIMENSION IDOF(1),D(6),IDO(6),XAX(3),YAX(3),ZAX(3)
      DIMENSION XAXR(3),YAXR(3),ZAXR(3),IUD(6)
C
      IRCORD = 0
      IDCORD = 0
      ICOMM = 0
      ITIME = 0
      ISTART = 0
      IEND = 0
      INTR = 0
C
      IF (N .NE. 0) GO TO 110
  100 CONTINUE
      READ (LDEV11, '(A80)', END=210) BUFFER
  110 CONTINUE
      CALL COMPRO (N, NVAR)
C
  120 CONTINUE
      IF (COMM.EQ.'NODE' .OR. COMM.EQ.'NODES') THEN
          IF (ITIME .EQ. 1) THEN
              ASSIGN 130 TO NEXT
              GO TO 180
          ENDIF
  130     CONTINUE
          READ (BUFF, *, END=160) ISTART
          ITIME = 1
          IEND = ISTART
          INTR = 1
          DO 140 K1 = 1, NNDF
              D(K1) = 0.
              IUD(K1) = 0
              IDO(K1) = 0
  140     CONTINUE
C
C ---- IDCORD is the flag for the translation coordinate definition.
C             = 0; a local coordinate system is not defined
C             = 1; a local coordinate system is defined
C ---- IRCORD is the flag for the rotational coordinate definition.
C             = 0; a local coordinate system is not defined
C             = 1; a local coordinate system is defined
C
          IRCORD = 0
          IDCORD = 0
```

```
C
      ELSE IF (COMM .EQ. 'TO') THEN
          READ (BUFF, *, END=160) IEND
      ELSE IF (COMM .EQ. 'BY') THEN
          READ (BUFF, *, END=160) INTR
      ELSE IF (COMM .EQ. 'DX') THEN
          READ (BUFF, *, END=160) D(1)
          IF (D(1) .EQ. 0.) THEN
              IDO(1) = 1
          ELSE
              IDO(1) = -1
          ENDIF
      ELSE IF (COMM .EQ. 'DY') THEN
          READ (BUFF, *, END=160) D(2)
          IF (D(2) .EQ. 0.) THEN
              IDO(2) = 1
          ELSE
              IDO(2) = -1
          ENDIF
      ELSE IF (COMM .EQ. 'DZ') THEN
          READ (BUFF, *, END=160) D(3)
          IF (D(3) .EQ. 0.) THEN
              IDO(3) = 1
          ELSE
              IDO(3) = -1
          ENDIF
      ELSE IF (COMM .EQ. 'RX') THEN
          READ (BUFF, *, END=160) D(4)
          IF (D(4) .EQ. 0.) THEN
              IDO(4) = 1
          ELSE
              IDO(4) = -1
          ENDIF
      ELSE IF (COMM .EQ. 'RY') THEN
          READ (BUFF, *, END=160) D(5)
          IF (D(5) .EQ. 0.) THEN
              IDO(5) = 1
          ELSE
              IDO(5) = -1
          ENDIF
      ELSE IF (COMM .EQ. 'RZ') THEN
          READ (BUFF, *, END=160) D(6)
          IF (D(6) .EQ. 0.) THEN
              IDO(6) = 1
          ELSE
              IDO(6) = -1
          ENDIF
      ELSE IF (COMM .EQ. 'HDX') THEN
          READ (BUFF, *, END=160) IUD(1)
      ELSE IF (COMM .EQ. 'HDY') THEN
          READ (BUFF, *, END=160) IUD(2)
      ELSE IF (COMM .EQ. 'HDZ') THEN
          READ (BUFF, *, END=160) IUD(3)
      ELSE IF (COMM .EQ. 'HRX') THEN
          READ (BUFF, *, END=160) IUD(4)
      ELSE IF (COMM .EQ. 'HRY') THEN
          READ (BUFF, *, END=160) IUD(5)
      ELSE IF (COMM .EQ. 'HRZ') THEN
          READ (BUFF, *, END=160) IUD(6)
      ELSE IF (COMM .EQ. 'END') THEN
          ICOMM = 0
          ASSIGN 150 TO NEXT
          GO TO 180
```

```
  150     CONTINUE
          RETURN
       ELSE IF (COMM .EQ. 'COORDI') THEN
          CALL IOCORD (IDIM, N, ICOMM, LDEV11, IOUT, XAX, YAX, ZAX)
          XAXR(1) = XAX(1)
          YAXR(1) = YAX(1)
          ZAXR(1) = ZAX(1)
          XAXR(2) = XAX(2)
          YAXR(2) = YAX(2)
          ZAXR(2) = ZAX(2)
          XAXR(3) = XAX(3)
          YAXR(3) = YAX(3)
          ZAXR(3) = ZAX(3)

C
          IRCORD = 1
          IDCORD = 1
          IF (ICOMM .EQ. 1) THEN
              GO TO 120
          ELSE IF (ICOMM .EQ. 2) THEN
              ASSIGN 210 TO NEXT
              GO TO 180
          ELSE
              GO TO 100
          ENDIF
       ELSE IF (COMM .EQ. 'DCOORD') THEN
          CALL IOCORD (IDIM, N, ICOMM, LDEV11, IOUT, XAX, YAX, ZAX)
          IDCORD = 1
          IF (ICOMM .EQ. 1) THEN
              GO TO 120
          ELSE IF (ICOMM .EQ. 2) THEN
              ASSIGN 210 TO NEXT
              GO TO 180
          ELSE
              GO TO 100
          ENDIF
       ELSE IF (COMM .EQ. 'RCOORD') THEN
          CALL IOCORD (IDIM, N, ICOMM, LDEV11, IOUT, XAXR, YAXR, ZAXR)
          IRCORD = 1
          IF (ICOMM .EQ. 1) THEN
              GO TO 120
          ELSE IF (ICOMM .EQ. 2) THEN
              ASSIGN 210 TO NEXT
              GO TO 180
          ELSE
              GO TO 100
          ENDIF
       ELSE IF (COMM .EQ. 'COMMEN') THEN
          GO TO 100
       ELSE
          ICOMM = 1
          ASSIGN 210 TO NEXT
          GO TO 180
       ENDIF
C
       GO TO 170
  160  CONTINUE
       CALL ERRORS (3, 1, 'IODISP')
  170  CONTINUE
       IF (N .NE. 0) THEN
          GO TO 110
       ELSE
          GO TO 100
```

```
      ENDIF
C
  180 CONTINUE
      DO 200 K1 = ISTART, IEND, INTR
C
C ---- Identify the node as a support
C
         ISPB(K1) = IBSET(ISPB(K1),18)
         ID1 = NNDF*(K1-1)
         DO 190 IDIR = 1, NNDF
             ID = ID1 + IDIR
             IF (D(IDIR) .NE. 0.) U(ID) = D(IDIR)
             IF (IUD(IDIR) .NE. 0.) IUDOF(ID) = IUD(IDIR)
             IF (IUD(IDIR) .EQ. 0.) IUDOF(ID) = 3
             IF (IDO(IDIR) .NE. 0) IDOF(ID) = IDO(IDIR)
  190    CONTINUE
C
C ----- ISPB( k ) identifies whether any coordinate systems have been
C       defined for the nodal displacements and rotations. It also
C       stores information on the sign of the third direction cosine
C       of each coordinate axis.
C       BIT 0    local coordinate system for translations
C       BIT 1    local coordinate system for rotations is defined
C       BIT 2    0; shell rotations in local shell coordinate system
C                1; shell rotations in global coordinate system.
C       BIT 3    sign bit for the third direction cosine of the
C                local translation X-axis (0=+,1=-).
C       BIT 4    sign bit for the third direction cosine of the
C                local translation Y-axis (0=+,1=-).
C       BIT 5    sign bit for the third direction cosine of the
C                local rotation X-axis (0=+,1=-).
C       BIT 6    sign bit for the third direction cosine of the
C                local rotation Y-axis (0=+,1=-).
C       BIT 7    sign bit for the third direction cosine of the
C                shell rotation Z-axis (0=+,1=-).
C
         IF (IDCORD .EQ. 1) THEN
             ISP = BTEST(ISPB(K1),0)
             IF (ISP) THEN
                 CALL ERRORS (4, 0, 'IODISP')
             ELSE
                 ISPB(K1) = ISPB(K1) + 1
             ENDIF
             XAXIS(1,K1) = XAX(1)
             YAXIS(1,K1) = YAX(1)
             XAXIS(2,K1) = XAX(2)
             YAXIS(2,K1) = YAX(2)
C
C ---- set the sign bits for the third direction cosines of the
C      of the local translation x and y axes.
C
             IF (XAX(3) .LT. 0.) ISPB(K1) = ISPB(K1) + 8
             IF (YAX(3) .LT. 0.) ISPB(K1) = ISPB(K1) + 16
         ENDIF
C
         IF (IRCORD .EQ. 1) THEN
             ISP = BTEST(ISPB(K1),1)
             IF (ISP) THEN
                 CALL ERRORS (4, 0, 'IODISP')
             ELSE
                 ISPB(K1) = ISPB(K1) + 2
             ENDIF
             XAXIS(3,K1) = XAXR(3)
```

```
            YAXIS(3,K1) = YAXR(3)
            XAXIS(4,K1) = XAXR(4)
            YAXIS(4,K1) = YAXR(4)
C
C ---- set the sign bits for the third direction cosines of the
C        of the local rotation x and y axes.
C
            IF (XAXR(3) .LT. O.) ISPB(K1) = ISPB(K1) + 32
            IF (YAXR(3) .LT. O.) ISPB(K1) = ISPB(K1) + 64
         ENDIF
C
  200 CONTINUE
      GO TO NEXT
C
  210 CONTINUE
      RETURN
      END
C
C ====================== I O C O R D ============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE IOCORD(IDIM,N,ICOMM,LDEV11,IOUT,XAX,YAX,ZAX)
C
C ==================================================================
C I
C I   P R O G R A M:
C I
C I   IOCORD reads and process the local coordinate system
C I   definitions.
C I
C I
C I   O N   E N T R Y:
C I
C I   IDIM   = physical dimension of the problem (i.e. 2D or 3D)
C I   N      = 0; buffer contains no additional commands
C I            1; Buffer containes at least one additional command
C I   LDEV11 = unit number for the input file
C I   IOUT   = unit number for the output file
C I
C I
C I   O N   R E T U R N:
C I
C I   N      = 0; buffer contains no additional commands
C I            1; Buffer containes at least one additional command
C I
C I   ICOMM  = identifies the action to be taken when control is
C I            returned to the calling routine .
C I            =0; read the next input line
C I            =1; The current command was not resolved by IODIS.
C I                Calling routine should process this command
C I            =2; end of file is reached, do not try to read any
C I                more lines.
C I
C I   XAX(i) = single percision direction cosines of the X-axis
C I
C I   YAX(i) = single percision direction cosines of the Y-axis
C I
C I   ZAX(i) = single percision direction cosines of the Z-axis
C I
C =================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
```

```
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C COMPRO    ERRORS    UNITV     DIRVEC    CROSS     DOTPRO
C
C************************************************************
      CHARACTER*80 BUFFER,BUFF
      CHARACTER*6 COMM
      REAL*4 XYZ,PERM,XAX,YAX,ZAX
      COMMON/INPUT3/XYZ(3,10000)
      COMMON/COMP2/COMM,BUFFER,BUFF
      DIMENSION X1(3),Y1(3),Z1(3),XZ(3),XY(3),YZ(3)
      DIMENSION VECT(3,6),PERM(3,3),ISTART(6),IEND(6),ID(6)
      DIMENSION COORD1(3,6),COORD2(3,6),XAX(3),YAX(3),ZAX(3)
      EQUIVALENCE (VECT(1,1),X1(1)),(VECT(1,2),Y1(1)),(VECT(1,3),Z1(1))
      EQUIVALENCE (VECT(1,4),XY(1)),(VECT(1,5),YZ(1)),(VECT(1,6),XZ(1))
      DATA PERM/0.,-1.,1.,1.,0.,-1.,-1.,1.,0./
C
      ICOMM = 0
      ID3 = 0
      IVEC = 0
      CST = 3.141592653589793D0/180.D0
C
      DO 100 K2 = 1, 6
          ID(K2) = 0
          ISTART(K2) = 0
          IEND(K2) = 0
          VECT(1,K2) = 0.D0
          VECT(2,K2) = 0.D0
          VECT(3,K2) = 0.D0
  100 CONTINUE
C
C ---- IVEC = 0; when vectors are defined by components
C            GE 0; when vectors are defined by FROM n TO m option
C
      IF (N .NE. 0) GO TO 120
  110 CONTINUE
      READ (LDEV11, '(A80)', END=130) BUFFER
  120 CONTINUE
      CALL COMPRO (N, NVAR)
C
      IF (IVEC .GT. 0) THEN
          IF (COMM .EQ. 'FROM') THEN
              IF (NVAR .EQ. 1) THEN
                  READ (BUFF, *) ISTART(IVEC)
              ELSE IF (NVAR.EQ.3 .OR. NVAR.EQ.IDIM) THEN
                  READ (BUFF, *) (COORD1(K1,IVEC), K1 = 1, NVAR)
                  ISTART(IVEC) = -1
              ELSE IF (NVAR .EQ. 0) THEN
                  CALL ERRORS (3, 1, 'IOCORD')
              ENDIF
          ELSE IF (COMM .EQ. 'TO') THEN
              IF (NVAR .EQ. 1) THEN
                  READ (BUFF, *) IEND(IVEC)
              ELSE IF (NVAR.EQ.3 .OR. NVAR.EQ.IDIM) THEN
                  READ (BUFF, *) (COORD2(K1,IVEC), K1 = 1, NVAR)
                  IEND(IVEC) = -1
              ELSE IF (NVAR .EQ. 0) THEN
                  CALL ERRORS (3, 1, 'IOCORD')
              ENDIF
              IVEC = 0
```

```
    ELSE
        IVEC = 0
        CALL ERRORS (7, 1, 'IOCORD')
    ENDIF
ELSE IF (COMM.EQ.'RECTAN' .OR. COMM.EQ.'CARTES') THEN
ELSE IF (COMM .EQ. 'CYLIND') THEN
ELSE IF (COMM .EQ. 'SPHERI') THEN
ELSE IF (COMM.EQ.'XVECTO' .OR. COMM.EQ.'XAXIS') THEN
    IVEC = 0
    ID(1) = 1
    IF (NVAR.EQ.IDIM .OR. NVAR.EQ.3) THEN
        READ (BUFF, *) (X1(K1), K1 = 1, IDIM)
    ELSE IF (NVAR .EQ. 0) THEN
        IVEC = 1
    ELSE
        CALL ERRORS (7, 1, 'IOCORD')
    ENDIF
ELSE IF (COMM.EQ.'YVECTO' .OR. COMM.EQ.'YAXIS') THEN
    IVEC = 0
    ID(2) = 1
    IF (NVAR.EQ.IDIM .OR. NVAR.EQ.3) THEN
        READ (BUFF, *) (Y1(K1), K1 = 1, IDIM)
    ELSE IF (NVAR .EQ. 0) THEN
        IVEC = 2
    ELSE
        CALL ERRORS (7, 1, 'IOCORD')
    ENDIF
ELSE IF (COMM.EQ.'ZVECTO' .OR. COMM.EQ.'ZAXIS') THEN
    IVEC = 0
    ID(3) = 1
    IF (NVAR.EQ.IDIM .OR. NVAR.EQ.3) THEN
        READ (BUFF, *) (Z1(K1), K1 = 1, IDIM)
    ELSE IF (NVAR .EQ. 0) THEN
        IVEC = 3
    ELSE
        CALL ERRORS (7, 1, 'IOCORD')
    ENDIF
ELSE IF (COMM .EQ. 'XYVECT') THEN
    IVEC = 0
    ID(4) = 1
    IF (NVAR.EQ.IDIM .OR. NVAR.EQ.3) THEN
        READ (BUFF, *) (XY(K1), K1 = 1, IDIM)
    ELSE IF (NVAR .EQ. 0) THEN
        IVEC = 4
    ELSE
        CALL ERRORS (7, 1, 'IOCORD')
    ENDIF
ELSE IF (COMM .EQ. 'XZVECT') THEN
    IVEC = 0
    ID(6) = 1
    IF (NVAR.EQ.IDIM .OR. NVAR.EQ.3) THEN
        READ (BUFF, *) (XZ(K1), K1 = 1, IDIM)
    ELSE IF (NVAR .EQ. 0) THEN
        IVEC = 6
    ELSE
        CALL ERRORS (7, 1, 'IOCORD')
    ENDIF
ELSE IF (COMM .EQ. 'YZVECT') THEN
    IVEC = 0
    ID(5) = 1
    IF (NVAR.EQ.IDIM .OR. NVAR.EQ.3) THEN
        READ (BUFF, *) (YZ(K1), K1 = 1, IDIM)
    ELSE IF (NVAR .EQ. 0) THEN
```

```
              IVEC = 5
          ELSE
              CALL ERRORS (7, 1, 'IOCORD')
          ENDIF
      ELSE IF (COMM .EQ. 'COMMEN') THEN
          GO TO 110
      ELSE
          ICOMM = 1
          GO TO 140
      ENDIF
C
      IF (N .NE. 0) THEN
          GO TO 120
      ELSE
          GO TO 110
      ENDIF
C
  130 CONTINUE
      ICOMM = 2
  140 CONTINUE
      ID1 = 0
      ID2 = 0
      DO 150 K1 = 1, 6
          IF (ID1 .EQ. 0) THEN
              IF (ID(K1) .EQ. 1) ID1 = K1
          ELSE IF (ID2 .EQ. 0) THEN
              IF (ID(K1) .EQ. 1) ID2 = K1
          ELSE
              GO TO 160
          ENDIF
  150 CONTINUE
C
  160 CONTINUE
      IDD = ID1
      DO 170 K1 = 1, 2
          IF (ISTART(IDD) .NE. 0) THEN
              IF (ISTART(IDD) .GT. 0) THEN
                  COORD1(1,IDD) = XYZ(1,ISTART(IDD))
                  COORD1(2,IDD) = XYZ(2,ISTART(IDD))
                  COORD1(3,IDD) = XYZ(3,ISTART(IDD))
              ENDIF
C
              IF (IEND(IDD) .GT. 0) THEN
                  COORD2(1,IDD) = XYZ(1,IEND(IDD))
                  COORD2(2,IDD) = XYZ(2,IEND(IDD))
                  COORD2(3,IDD) = XYZ(3,IEND(IDD))
              ENDIF
              VECT(1,IDD) = COORD2(1,IDD) - COORD1(1,IDD)
              VECT(2,IDD) = COORD2(2,IDD) - COORD1(2,IDD)
              VECT(3,IDD) = COORD2(3,IDD) - COORD1(3,IDD)
          ENDIF
          IDD = ID2
  170 CONTINUE
C
      IF (ID1.LT.3 .AND. ID2.EQ.0) THEN
          CALL UNITV (VECT(1,ID1),3)
          IF (ID1 .EQ. 1) THEN
              CALL DIRVEC (VECT(1,2),VECT(1,3),VECT(1,1))
          ELSE IF (ID1 .EQ. 2) THEN
              CALL DIRVEC (VECT(1,3),VECT(1,1),VECT(1,2))
          ELSE IF (ID1 .EQ. 3) THEN
              CALL DIRVEC (VECT(1,1),VECT(1,2),VECT(1,3))
```

```
            ELSE
                CALL ERRORS (5, 1, 'IOCORD')
            ENDIF
        ELSE IF (ID1 .GT. 3.) THEN
            CALL ERRORS (5, 1, 'IOCORD')
        ELSE IF (ID1 .EQ. ID2) THEN
            CALL ERRORS (5, 1, 'IOCORD')
        ELSE IF (ID1 .LT. 3) THEN
            IF (ID2 .LT. 3) THEN
                GO TO 180
            ELSE IF (ID1 .EQ. 1) THEN
                IF (ID2 .EQ. 5) THEN
                    Y1(1) = YZ(1)
                    Y1(2) = YZ(2)
                    Y1(3) = YZ(3)
                    ID2 = 2
                ELSE IF (ID2 .EQ. 4) THEN
                    CALL CROSS (X1, XY, Z1)
                    ID2 = 3
                ELSE IF (ID2 .EQ. 6) THEN
                    CALL CROSS (XZ, X1, Y1)
                    ID2 = 2
                ENDIF
            ELSE IF (ID1 .EQ. 2) THEN
                IF (ID2 .EQ. 6) THEN
                    X1(1) = XZ(1)
                    X1(2) = XZ(2)
                    X1(3) = XZ(3)
                    ID2 = 1
                ELSE IF (ID2 .EQ. 4) THEN
                    CALL CROSS (XY, Y1, Z1)
                    ID2 = 3
                ELSE IF (ID2 .EQ. 5) THEN
                    CALL CROSS (Y1, YZ, X1)
                    ID2 = 1
                ENDIF


            ELSE IF (ID1 .EQ. 3) THEN
                IF (ID2 .EQ. 4) THEN
                    X1(1) = XY(1)
                    X1(2) = XY(2)
                    X1(3) = XY(3)
                    ID2 = 1
                ELSE IF (ID2 .EQ. 5) THEN
                    CALL CROSS (YZ, Z1, X1)
                    ID2 = 1
                ELSE IF (ID2 .EQ. 6) THEN
                    CALL CROSS (Z1, XZ, Y1)
                    ID2 = 2
                ENDIF
            ENDIF
C
  180       CONTINUE
            CALL UNITV (VECT(1,ID1),3)
            CALL UNITV (VECT(1,ID2),3)
            DOT = DOTPRO(VECT(1,ID1),VECT(1,ID2),3)
            IF (DABS(DOT) .GT. 1.0D-4) CALL ERRORS (6, 1, 'IOCORD')
C
            DO 190 K1 = 1, 3
                IF (K1.NE.ID1 .AND. K1.NE.ID2) ID3 = K1
  190       CONTINUE
C
            CALL CROSS (VECT(1,ID1),VECT(1,ID2),VECT(1,ID3))
```

```
         CONST = PERM(ID1,ID2)
         VECT(1,ID3) = VECT(1,ID3)*CONST
         VECT(2,ID3) = VECT(2,ID3)*CONST
         VECT(3,ID3) = VECT(3,ID3)*CONST
C
         XAX(1) = X1(1)
         YAX(1) = Y1(1)
         ZAX(1) = Z1(1)
         XAX(2) = X1(2)
         YAX(2) = Y1(2)
         ZAX(2) = Z1(2)
         XAX(3) = X1(3)
         YAX(3) = Y1(3)
         ZAX(3) = Z1(3)
C
      ENDIF
C
  200 CONTINUE
      RETURN
      END
C
C ======================== I O L O A D ===============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE IOLOAD(IDIM,N,NNDF,ICOMM,LDEV11,IOUT)
C
C ====================================================================
C I
C I   P R O G R A M:
C I
C I   IOLOAD reads and stores the structural loads in either the
C I   global or the local coordinate system.
C I
C I
C I   O N   E N T R Y:
C I
C I   IDIM   = physical dimension of the problem (i.e. 2D or 3D)
C I   NNDF   = number of nodal degrees of freedom
C I   N      = 0; buffer contains no additional commands
C I             1; Buffer containes at least one additional command
C I   LDEV11 = unit number for the input file
C I   IOUT   = unit number for the output file
C I
C I
C I   O N   R E T U R N:
C I
C I   N      = 0; buffer contains no additional commands
C I             1; Buffer containes at least one additional command
C I
C I   ICOMM  = identifies the action to be taken when control is
C I             returned to the calling routine .
C I             =0; read the next input line
C I             =1; The current command was not resolved by IODIS.
C I                 Calling routine should process this command
C I             =2; end of file is reached, do not try to read any
C I                 more lines.
C I
C ==================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C**************************************************************
```

```
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C COMPRO    IOCORD    ERRORS
C
C*************************************************************
      CHARACTER*80 BUFFER,BUFF
      CHARACTER*6 COMM
      REAL*4 XAX,YAX,ZAX,XAXR,YAXR,ZAXR
      COMMON/COMP2/COMM,BUFFER,BUFF
      COMMON/SAFE2/R(60000),IDOF(60000),JDIAG(60000)
      COMMON/SAFE4/IRDOF(60000),IUDOF(60000)
      DIMENSION P(6),XAX(3),YAX(3),ZAX(3),XAXR(3),YAXR(3),ZAXR(3),IRD(6)
C
      ITIME = 0
      IRCORD = 0
      IDCORD = 0
      ICOMM = 0
      ISTART = 0
      IEND = 0
      INTR = 0
C
      IF (N .NE. 0) GO TO 110
  100 CONTINUE
      READ (LDEV11, '(A80)', END=190) BUFFER
  110 CONTINUE
      CALL COMPRO (N, NVAR)
C
  120 CONTINUE
      IF (COMM.EQ.'NODE' .OR. COMM.EQ.'NODES') THEN
          IF (ITIME .EQ. 1) THEN
              ASSIGN 130 TO NEXT
              GO TO 160
          ENDIF
  130     CONTINUE
          READ (BUFF, *, END=200) ISTART
          ITIME = 1
          IEND = ISTART
          INTR = 1
C
C ---- IDCORD is the flag for the translation coordinate definition.
C              = 0; a local coordinate system is not defined
C              = 1; a local coordinate system is defined
C ----> IRCORD is the flag for the rotational coordinate definition.
C              = 0; a local coordinate system is not defined
C              = 1; a local coordinate system is defined
C
          IRCORD = 0
          IDCORD = 0
          DO 140 K1 = 1, NNDF
              IRD(K1) = 0.
              P(K1) = 0.
  140     CONTINUE
      ELSE IF (COMM .EQ. 'TO') THEN
          READ (BUFF, *, END=200) IEND
      ELSE IF (COMM .EQ. 'BY') THEN
          READ (BUFF, *, END=200) INTR
      ELSE IF (COMM .EQ. 'FX') THEN
          READ (BUFF, *, END=200) P(1)
      ELSE IF (COMM .EQ. 'FY') THEN
          READ (BUFF, *, END=200) P(2)
      ELSE IF (COMM .EQ. 'FZ') THEN
          READ (BUFF, *, END=200) P(3)
```

```
      ELSE IF (COMM .EQ. 'MX') THEN
          READ (BUFF, *, END=200) P(4)
      ELSE IF (COMM .EQ. 'MY') THEN
          READ (BUFF, *, END=200) P(5)
      ELSE IF (COMM .EQ. 'MZ') THEN
          READ (BUFF, *, END=200) P(6)
      ELSE IF (COMM .EQ. 'HFX') THEN
          READ (BUFF, *, END=200) IRD(1)
      ELSE IF (COMM .EQ. 'HFY') THEN
          READ (BUFF, *, END=200) IRD(2)
      ELSE IF (COMM .EQ. 'HFZ') THEN
          READ (BUFF, *, END=200) IRD(3)
      ELSE IF (COMM .EQ. 'HMX') THEN
          READ (BUFF, *, END=200) IRD(4)
      ELSE IF (COMM .EQ. 'HMY') THEN
          READ (BUFF, *, END=200) IRD(5)
      ELSE IF (COMM .EQ. 'HMZ') THEN
          READ (BUFF, *, END=200) IRD(6)
      ELSE IF (COMM .EQ. 'COORDI') THEN
          CALL IOCORD (IDIM, N, ICOMM, LDEV11, IOUT, XAX, YAX, ZAX)
          XAXR(1) = XAX(1)
          YAXR(1) = YAX(1)
          ZAXR(1) = ZAX(1)
          XAXR(2) = XAX(2)
          YAXR(2) = YAX(2)
          ZAXR(2) = ZAX(2)
          XAXR(3) = XAX(3)
          YAXR(3) = YAX(3)
          ZAXR(3) = ZAX(3)
C
          IRCORD = 1
          IDCORD = 1
          IF (ICOMM .EQ. 1) THEN
              GO TO 120
          ELSE IF (ICOMM .EQ. 2) THEN
              ASSIGN 190 TO NEXT
              GO TO 160
          ELSE
              GO TO 100
          ENDIF
      ELSE IF (COMM .EQ. 'DCOORD') THEN
          CALL IOCORD (IDIM, N, ICOMM, LDEV11, IOUT, XAX, YAX, ZAX)
          IDCORD = 1
          IF (ICOMM .EQ. 1) THEN
              GO TO 120
          ELSE IF (ICOMM .EQ. 2) THEN
              ASSIGN 190 TO NEXT
              GO TO 160
          ELSE
              GO TO 100
          ENDIF
      ELSE IF (COMM .EQ. 'RCOORD') THEN
          CALL IOCORD (IDIM, N, ICOMM, LDEV11, IOUT, XAXR, YAXR, ZAXR)
          IRCORD = 1
          IF (ICOMM .EQ. 1) THEN
              GO TO 120
          ELSE IF (ICOMM .EQ. 2) THEN
              ASSIGN 190 TO NEXT
              GO TO 160
          ELSE
              GO TO 100
          ENDIF
      ELSE IF (COMM .EQ. 'END') THEN
```

```
          ICOMM = 0
          ASSIGN 150 TO NEXT
          GO TO 160
  150     CONTINUE
          RETURN
      ELSE IF (COMM .EQ. 'COMMEN') THEN
          GO TO 100
      ELSE
          ICOMM = 1
          ASSIGN 150 TO NEXT
          GO TO 160
      ENDIF
C
      IF (N .NE. 0) THEN
          GO TO 110
      ELSE
          GO TO 100
      ENDIF
C
  160 CONTINUE
      IF (IDCORD .EQ. 1) THEN
          CST1 = XAX(1)*P(1) + YAX(1)*P(2) + ZAX(1)*P(3)
          CST2 = XAX(2)*P(1) + YAX(2)*P(2) + ZAX(2)*P(3)
          CST3 = XAX(3)*P(1) + YAX(3)*P(2) + ZAX(3)*P(3)
          P(1) = CST1
          P(2) = CST2
          P(3) = CST3
      ENDIF
C
      IF (IRCORD .EQ. 1) THEN
          CST1 = XAXR(1)*P(4) + YAXR(1)*P(5) + ZAXR(1)*P(6)
          CST2 = XAXR(2)*P(4) + YAXR(2)*P(5) + ZAXR(2)*P(6)
          CST3 = XAXR(3)*P(4) + YAXR(3)*P(5) + ZAXR(3)*P(6)
          P(4) = CST1
          P(5) = CST2
          P(6) = CST3
      ENDIF
C
      DO 180 K1 = ISTART, IEND, INTR
          ID1 = NNDF*(K1-1)
          DO 170 IDIR = 1, NNDF
              ID = ID1 + IDIR
              IF (IRD(IDIR) .NE. 0.) IRDOF(ID) = IRD(IDIR)
              IF (IRD(IDIR) .EQ. 0.) IRDOF(ID) = 3
              IF (P(IDIR) .NE. 0.) R(ID) = P(IDIR)
  170     CONTINUE
  180 CONTINUE
C
      GO TO NEXT
C
  190 CONTINUE
      ICOMM = 2
      RETURN
  200 CONTINUE
      CALL ERRORS (3, 1, 'IOLOAD')
      STOP
      END
C
C ======================== E L E M E N ===============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE ELEMEN(N,ICOMM,LDEV11,IOUT,NNELEM)
C
```

```
C ================================================================
C I
C I   P R O G R A M:
C I
C I   ELEMEN reads and stores physical properties of the elements
C I
C I
C I   O N   E N T R Y:
C I
C I   IDIM   = physical dimension of the problem (i.e. 2D or 3D)
C I   N      = 0; buffer contains no additional commands
C I            1; Buffer containes at least one additional command
C I   LDEV11 = unit number for the input file
C I   IOUT   = unit number for the output file
C I
C I
C I   O N   R E T U R N:
C I
C I   N      = 0; buffer contains no additional commands
C I            1; Buffer containes at least one additional command
C I
C I   ICOMM  = identifies the action to be taken when control is
C I            returned to the calling routine .
C I            =0; read the next input line
C I            =1; The current command was not resolved by IODIS.
C I                Calling routine should process this command
C I            =2; end of file is reached, do not try to read any
C I                more lines.
C I
C I   NNELEM = temperary storage for number of nodes in the elements.
C I
C ===============================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C COMPRO    ERRORS
C
C***********************************************************
      CHARACTER*80 BUFFER,BUFF
      CHARACTER*6 COMM
c changes 7/18
c***  REAL*4 ETHICK
c changes 7/18
      REAL*4 THICK,T1,T2,T3,T4
      COMMON/INPUTA/INFOEL(6,5000)
      COMMON/COMP2/COMM,BUFFER,BUFF
      DIMENSION NNELEM( * )
c changes 7/18
      EQUIVALENCE (T1,IT1),(T2,IT2),(T3,IT3),(T4,IT4)
c changes 7/18
C
C ---- read and generate element information
C
      IFLAG = 0
      ICOMM = 0
      LINES = 0
      LOCATE = 0
      ITF = 0
```

```
      READ (BUFF, *, END=160) ISTART
C
c changes 7/18
      T1 = 0.0
      T2 = 0.0
      T3 = 0.0
      T4 = 0.0
c changes 7/18
      IEND = ISTART
      INTR = 1
      MAT = 0
      NIPXI = 0
      NIPETA = 0
      NIPSI = 0
      INTCOD = 0
      THICK = 0.
      ITYPE = 0
      IF (N .NE. 0) GO TO 110
  100 CONTINUE
      READ (LDEV11, '(A80)', END=120) BUFFER
  110 CONTINUE
      CALL COMPRO (N, NVAR)
C
      IF (COMM .EQ. 'TO') THEN
          READ (BUFF, *, END=160) IEND
      ELSE IF (COMM .EQ. 'BY') THEN
          READ (BUFF, *, END=160) INTR
      ELSE IF (COMM .EQ. 'NIPXI') THEN
          READ (BUFF, *, END=160) NIPXI
      ELSE IF (COMM .EQ. 'NIPETA') THEN
          READ (BUFF, *, END=160) NIPETA
      ELSE IF (COMM .EQ. 'NIPZET') THEN
          READ (BUFF, *, END=160) NIPSI
      ELSE IF (COMM.EQ.'THICKN' .OR. COMM.EQ.'THICK') THEN
          READ (BUFF, *, END=160) THICK
          ITF = 30
          T1 = THICK
          T2 = THICK
          T3 = THICK
          T4 = THICK
      ELSE IF (COMM .EQ. 'T1') THEN
          READ (BUFF, *, END=160) T1
          ITF = ITF + 2
      ELSE IF (COMM .EQ. 'T2') THEN
          READ (BUFF, *, END=160) T2
          ITF = ITF + 4
      ELSE IF (COMM .EQ. 'T3') THEN
          READ (BUFF, *, END=160) T3
          ITF = ITF + 8
      ELSE IF (COMM .EQ. 'T4') THEN
          READ (BUFF, *, END=160) T4
          ITF = ITF + 16
      ELSE IF (COMM.EQ.'MATERI' .OR. COMM.EQ.'MAT') THEN
          READ (BUFF, *, END=160) MAT
          IF (MAT .GT. 7) CALL ERRORS (13, 1, 'ELEMEN')
      ELSE IF (COMM .EQ. 'TYPE') THEN
          READ (BUFF, *, END=160) ITYPE


      ELSE IF (COMM .EQ. 'COMMEN') THEN
          GO TO 100
      ELSE
          ICOMM = 1
          GO TO 130
```

```
      ENDIF
C
      IF (N .NE. 0) THEN
          GO TO 110
      ELSE
          GO TO 100
      ENDIF
C
C
C
C ---- Condensed data storage in array INFOEL.
C
C ---- Bit storage organization for INFOEL(1,elnum)
C
C       Bit Range      Max. Value       Description
C
C       0-2            7                material number
C       3-8            63               number of nodes in the element
C       9-17           511              element type number
C       18-20          7                analysis type flag
C       21-23          7                integration points in XI direc.
C       24-26          7                integration points in ETA direc.
C       27-30          7                integration points in ZETA dir.
C
C
C ---- Bit storage organization for INFOEL(2,elnum)
C
C       Bit Range      Max. Value       Description
C
C       0-7            255              integration code
C       8-14           128              no. of layers
C       15-16          3                flag for layering.
C       17-22          63               number of lines connecting nodes
C       23-30          255              starting position of line
C                                       connectivity in arrays ISTART &
C                                       IEND
C
C
C ---- Table of internal element type numbers
C
C       Type No.   No. of nodes   Description
C
C       204        4              linear isoparametric quadrilateral
C       208        8              quadratic isoparametric quadrilateral
C       209        9              Lagrangian quadratic isop. quad.
C       219        9              quadratic to qubic isop. quad.
C                  4              linear isoparametric brick or
C       308        8              quadratic shell
C       309        9              lagrangian quadratic shell
C       320        20             quadratic isoparametric brick
C
  120 CONTINUE
      ICOMM = 2
  130 CONTINUE
      NNEL = 0
      IF (ITYPE .NE. 0) THEN
          ID = ITYPE/1000
          ID1 = ITYPE - ID*1000
          IFLAG = ID1/100
          NNEL = ID1 - IFLAG*100
          ITYPE = ID*100 + NNEL
          IF (ITYPE .LT. 300) THEN
              IF (NNEL .EQ. 4) THEN
```

```
                LOCATE = 1
                LINES = 4
            ELSE IF (NNEL .EQ. 5) THEN
                LOCATE = 5
                LINES = 5
            ELSE IF (ITYPE.EQ.208 .OR. ITYPE.EQ.209) THEN
                LOCATE = 10
                LINES = 8
            ELSE IF (ITYPE .EQ. 219) THEN
                NNEL = 9
                LOCATE = 54
                LINES = 9
            ELSE
                CALL ERRORS (12, 1, 'ELEMEN')
            ENDIF
        ELSE IF (ITYPE .GT. 300) THEN
            IF (IFLAG .EQ. 0) THEN
                IF (NNEL .EQ. 8) THEN
                    LOCATE = 18
                    LINES = 12
                ELSE IF (NNEL .EQ. 20) THEN
                    LOCATE = 30
                    LINES = 24
                ELSE
                    CALL ERRORS (12, 1, 'ELEMEN')
                ENDIF
            ELSE IF (IFLAG .EQ. 4) THEN
                IF (ITYPE.EQ.308 .OR. ITYPE.EQ.309) THEN
                    LOCATE = 10
                    LINES = 8
                ELSE IF (ITYPE .EQ. 304) THEN
                    LOCATE = 1
                    LINES = 4
                ELSE
                    CALL ERRORS (12, 1, 'ELEMEN')
                ENDIF
            ELSE
                CALL ERRORS (12, 1, 'ELEMEN')
            ENDIF
        ELSE
            CALL ERRORS (12, 1, 'ELEMEN')
        ENDIF
      ENDIF
C
C ---- AND operations are used to clear the bits prior to storage
C        of the appropriate values.
C
      DO 140 K = ISTART, IEND, INTR
        IF (ITYPE .NE. 0) THEN
            INF1 = IAND(INFOEL(1,K),2145386503)
            INFOEL(1,K) = INF1 + NNEL*8 + ITYPE*512 + IFLAG*262144
            NNELEM(K) = NNEL
            INF2 = IAND(INFOEL(2,K),2139226111)
            INFOEL(2,K) = INF2 + 131072*LINES
            INF2 = IAND(INFOEL(2,K),8388607)
            INFOEL(2,K) = INF2 + 8388608*LOCATE
        ENDIF
        IF (MAT .NE. 0) THEN
            INF1 = IAND(INFOEL(1,K),2147483640)
            INFOEL(1,K) = INF1 + MAT
        ENDIF
        IF (NIPXI .NE. 0) THEN
            INFOEL(2,K) = IAND(INFOEL(2,K),2147483392)
```

```
              INF1 = IAND(INFOEL(1,K),2132803583)
              INFOEL(1,K) = INF1 + 2097152*NIPXI
          ENDIF
          IF (NIPETA .NE. 0) THEN
              INFOEL(2,K) = IAND(INFOEL(2,K),2147483392)
              INF1 = IAND(INFOEL(1,K),2030043135)
              INFOEL(1,K) = INF1 + 16777216*NIPETA
          ENDIF
          IF (NIPSI .NE. 0) THEN
              INFOEL(2,K) = IAND(INFOEL(2,K),2147483392)
              INF1 = IAND(INFOEL(1,K),134217727)
              INFOEL(1,K) = INF1 + 134217728*NIPSI
          ENDIF
          IF (INTCOD .NE. 0) THEN
              INF2 = IAND(INFOEL(2,K),2147483392)
              INFOEL(2,K) = INF2 + INTCOD
          ENDIF
          IF (BTEST(ITF,1)) INFOEL(3,K) = IT1
          IF (BTEST(ITF,2)) INFOEL(4,K) = IT2
          IF (BTEST(ITF,3)) INFOEL(5,K) = IT3
          IF (BTEST(ITF,4)) INFOEL(6,K) = IT4
c changes 7/18
  140 CONTINUE
C
  150 CONTINUE
      RETURN
  160 CONTINUE
      CALL ERRORS (3, 1, 'ELEMEN')
      END
C
C ======================== I O F R E E =============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE IOFREE(N,ICOMM,LDEV11,IOUT)
C
C ==================================================================
C I
C I   P R O G R A M:
C I
C I   IOFREE reads the freebody definitions
C I
C I
C I   O N   E N T R Y:
C I
C I   IDIM   = physical dimension of the problem (i.e. 2D or 3D)
C I   N      = 0; buffer contains no additional commands
C I             1; Buffer containes at least one additional command
C I   LDEV11 = unit number for the input file
C I   IOUT   = unit number for the output file
C I
C I
C I   O N   R E T U R N:
C I
C I   N      = 0; buffer contains no additional commands
C I             1; Buffer containes at least one additional command
C I
C I   ICOMM  = identifies the action to be taken when control is
C I             returned to the calling routine .
C I             =0; read the next input line
C I             =1; The current command was not resolved by IODIS.
C I                 Calling routine should process this command
C I             =2; end of file is reached, do not try to read any
C I                 more lines.
```

```
C I
C I
C ===================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C**********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C COMPRO    ERRORS
C
C**********************************************************
      CHARACTER*80 BUFFER,BUFF
      CHARACTER*6 COMM
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1              NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC1(10),LDCONT,INCPTR
      COMMON/COMP2/COMM,BUFFER,BUFF
      COMMON/FREEB/IFBODY(10000)
      DIMENSION IELEM(30)
C
C ---- read and generate freebody information
C
      ISTART = 0
      IEND = 0
      INTR = 0
      NVAR = 0
      ITIME = 0
      ICOMM = 0
      READ (BUFF, *, END=190) NUM
      NFREE = MAX0(NFREE,NUM)
C
      IF (N .NE. 0) GO TO 110
  100 CONTINUE
      READ (LDEV11, '(A80)', END=150) BUFFER
  110 CONTINUE
      CALL COMPRO (N, NVAR)
C
      IF (COMM.EQ.'ELEM' .OR. COMM.EQ.'ELEMEN') THEN
          IF (NVAR .EQ. 1) THEN
              IF (ITIME .EQ. 1) THEN
                  ASSIGN 120 TO NEXT
                  GO TO 160
              ENDIF
              ITIME = 1
C
  120         CONTINUE
              READ (BUFF, *, END=190) ISTART
              INTR = 1
              IEND = ISTART
          ELSE IF (NVAR .GT. 1) THEN
              IF (ITIME .EQ. 1) THEN
                  ASSIGN 130 TO NEXT
                  GO TO 160
              ENDIF
              ITIME = 0
  130         CONTINUE
              READ (BUFF, *, END=190) (IELEM(K1), K1 = 1, NVAR)
              DO 140 K1 = 1, NVAR
                  IFBODY(IELEM(K1)) = IBSET(IFBODY(IELEM(K1)),NUM)
  140         CONTINUE
```

```
            IF (N .NE. 0) THEN
               GO TO 110
            ELSE
               GO TO 100
            ENDIF
         ENDIF
      ELSE IF (COMM .EQ. 'TO') THEN
         READ (BUFF, *, END=190) IEND
         IF (ISTART .EQ. 0) CALL ERRORS (19, 1, 'IOFREE')
      ELSE IF (COMM .EQ. 'BY') THEN
         READ (BUFF, *, END=190) INTR
      ELSE IF (COMM .EQ. 'COMMEN') THEN
         GO TO 100
      ELSE
         ICOMM = 1
         ASSIGN 180 TO NEXT
         GO TO 160
      ENDIF
C
      IF (N .NE. 0) THEN
         GO TO 110
      ELSE
         GO TO 100
      ENDIF
C
C
C
C ---- Condensed data storage in array IFBODY.
C
C ---- Bit storage organization for IFBODY
C
C       Bit Range      Entity     Description
C
C       1-15           element    0; does not belong to freebody
C                                 1; belongs to freebody
C       16-30          nodes      0; does not belongs to freebody
C                                 1; belonds to freebody
C
C       For elements freebody number is the bit number.
C       For nodes freebody number is bit number minus 15.
C
  150 CONTINUE
      ICOMM = 2
      ASSIGN 180 TO NEXT
  160 CONTINUE
      DO 170 K1 = ISTART, IEND, INTR
         IFBODY(K1) = IBSET(IFBODY(K1),NUM)
  170 CONTINUE
      GO TO NEXT
C
  180 CONTINUE
      RETURN
  190 CONTINUE
      CALL ERRORS (3, 1, 'IOFREE')
      END
C
C ======================= I O M A T R ==============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE IOMATR(N,ICOMM,LDEV11,IOUT)
C
C ==================================================================
C I
```

```
C I   P R O G R A M:
C I
C I   IOMATR reads and stores the material properties.
C I
C I
C I   O N   E N T R Y:
C I
C I   N      = 0; buffer contains no additional commands
C I            1; Buffer containes at least one additional command
C I   LDEV11 = unit number for the input file
C I   IOUT   = unit number for the output file
C I
C I
C I   O N   R E T U R N:
C I
C I   N      = 0; buffer contains no additional commands
C I            1; Buffer containes at least one additional command
C I
C I   ICOMM  = identifies the action to be taken when control is
C I            returned to the calling routine .
C I            =0; read the next input line
C I            =1; the current command was not resolved by IOMATR.
C I                Calling routine should process this command
C I            =2; end of file is reached, do not try to read any
C I                more lines.
C I
C ===================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C COMPRO    ERRORS
C
C************************************************************
      CHARACTER*80 BUFFER,BUFF
      CHARACTER*6 COMM
      REAL*8 NUX,NUY,NUZ
      COMMON/INPUTF/MATYPE(10)
      COMMON/INPUT5/NUX(10),NUY(10),NUZ(10),EX(10),EY(10),EZ(10),
     1 P1X(10),P1Y(10),P1Z(10),P2X(10),P2Y(10),P2Z(10)
      COMMON/INPUT6/WGTX(10),WGTY(10),WGTZ(10)
      COMMON/COMP2/COMM,BUFFER,BUFF
C
      ICOMM = 0
C
      READ (BUFF, *, END=140) MATNUM
      IF (N .NE. 0) GO TO 110
  100 CONTINUE
      READ (LDEV11, '(A80)', END=120) BUFFER
  110 CONTINUE
      CALL COMPRO (N, NVAR)
C
      IF (COMM .EQ. 'NU') THEN
          READ (BUFF, *, END=140) NUX(MATNUM)
          NUY(MATNUM) = NUX(MATNUM)
          NUZ(MATNUM) = NUX(MATNUM)
      ELSE IF (COMM .EQ. 'E') THEN
          READ (BUFF, *, END=140) EX(MATNUM)
          EY(MATNUM) = EX(MATNUM)
```

```
          EZ(MATNUM) = EX(MATNUM)
      ELSE IF (COMM .EQ. 'WX') THEN
          READ (BUFF, *, END=140) WGTX(MATNUM)
      ELSE IF (COMM .EQ. 'WY') THEN
          READ (BUFF, *, END=140) WGTY(MATNUM)
      ELSE IF (COMM .EQ. 'WZ') THEN
          READ (BUFF, *, END=140) WGTZ(MATNUM)
      ELSE IF (COMM .EQ. 'TYPE') THEN
          READ (BUFF, *, END=140) MATYPE(MATNUM)
      ELSE IF (COMM .EQ. 'YIELD') THEN
          READ (BUFF, *, END=140) P1Z(MATNUM)
      ELSE IF (COMM.EQ.'ISOTRO' .OR. COMM.EQ.'ISOT') THEN
          READ (BUFF, *, END=140) P1Y(MATNUM)
      ELSE IF (COMM.EQ.'KINE' .OR. COMM.EQ.'KINEMA') THEN
          READ (BUFF, *, END=140) P1X(MATNUM)
      ELSE IF (COMM .EQ. 'COMMEN') THEN
          GO TO 100


      ELSE
          ICOMM = 1
          GO TO 130
      ENDIF
C
      IF (N .NE. 0) THEN
          GO TO 110
      ELSE
          GO TO 100
      ENDIF
C
  120 CONTINUE
      ICOMM = 2
  130 CONTINUE
      IF (MATYPE(MATNUM) .EQ. 0) THEN
          WRITE (BUFFER, *) 'MATERIAL', MATNUM
          CALL ERRORS (18, 1, 'IOMATR')
      ENDIF
C
      RETURN
  140 CONTINUE
      CALL ERRORS (3, 1, 'IOMATR')
      STOP
      END
C
C ====================== G R A P H X ============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE GRAPHX(N,ICOMM,LDEV11,IOUT)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C COMPRO
C
C************************************************************
      CHARACTER*80 BUFFER,BUFF
      CHARACTER*6 COMM
      REAL*4 FMAG,DMAG
      REAL*8 ANGLE,HIGHT
      COMMON/GRAPH3/XL,XR,YB,YT,ZF,D
      COMMON/GRAPH4/XVL,XVR,YVB,YVT,SX,SY
      COMMON/GRAPH5/FMAG,DMAG,HIGHT,ANGLE,NOLINE,ITHICK,NLINES
```

```
      COMMON/COMP2/COMM,BUFFER,BUFF
C
C          READ THE COMMAND LINE BUFFER
C
      ICOMM = 0
      IF (N .NE. 0) GO TO 110
  100 CONTINUE
      READ (LDEV11, '(A80)', END=120) BUFFER
  110 CONTINUE
      CALL COMPRO (N, NVAR)
      IF (N .EQ. 0) THEN
         ASSIGN 100 TO NEXT
      ELSE
         ASSIGN 110 TO NEXT
      ENDIF
C
      IF (COMM .EQ. 'FMAG') THEN
         READ (BUFF, *, END=130) FMAG
      ELSE IF (COMM .EQ. 'DMAG') THEN
         READ (BUFF, *, END=130) DMAG
      ELSE IF (COMM .EQ. 'WL') THEN
         READ (BUFF, *, END=130) XL
      ELSE IF (COMM .EQ. 'WR') THEN
         READ (BUFF, *, END=130) XR
      ELSE IF (COMM .EQ. 'WT') THEN
         READ (BUFF, *, END=130) YT
      ELSE IF (COMM .EQ. 'WB') THEN
         READ (BUFF, *, END=130) YB
      ELSE IF (COMM .EQ. 'VL') THEN
         READ (BUFF, *, END=130) XVL
      ELSE IF (COMM .EQ. 'VR') THEN
         READ (BUFF, *, END=130) XVR
      ELSE IF (COMM .EQ. 'VT') THEN
         READ (BUFF, *, END=130) YVT
      ELSE IF (COMM .EQ. 'VB') THEN
         READ (BUFF, *, END=130) YVB
      ELSE IF (COMM.EQ.'THICK' .OR. COMM.EQ.'THICKN') THEN
         READ (BUFF, *, END=130) ITHICK
      ELSE IF (COMM .EQ. 'CONTOU') THEN
         NOLINE = 1
      ELSE IF (COMM .EQ. 'HIGHT') THEN
         READ (BUFF, *, END=130) HIGHT
      ELSE IF (COMM .EQ. 'ANGLE') THEN
         READ (BUFF, *, END=130) ANGLE
      ELSE IF (COMM .EQ. 'COMMEN') THEN
         GO TO 100
      ELSE IF (COMM .EQ. 'END') THEN
         RETURN
      ELSE
         ICOMM = 1
         RETURN
      ENDIF
      GO TO NEXT
  120 CONTINUE
      ICOMM = 2
      RETURN
  130 CONTINUE
      STOP
      END
C
C ======================= I O O U T ===============================
C
      SUBROUTINE IOOUT(N,ICOMM,LDEV11)
```

```
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C COMPRO    ERRORS
C
C************************************************************
      CHARACTER*80 BUFFER,BUFF
      CHARACTER*6 COMM
      COMMON/COMP2/COMM,BUFFER,BUFF
      COMMON/INPUTG/IFLAG3,IOINTR,IFPLOT
      COMMON/OUTPT1/IOFLAG,IFCAED
      COMMON/OUTPT2/IOEL(5000),IONOD(10000)
C
C ---- Description of the output modules and the corresponding bit
C        flag position in the IOFLAG integer variable
C
C        file    bit  description
C
C        OUT1    1    output element stresses at the integration points
C        OUT2    2    output element strains at the integration points
C        OUT3    3    output element stresses at the element nodes
C        OUT4    4    output element strains at the element nodes
C        OUT5    5    output average stresses at the nodes
C        OUT6    6    output average strains at the nodes
C        OUT7    7    output displacements at the nodes
C        OUT8    8    output nodal equilibrium forces at the nodes
C        OUT9    9    output nodal equilibrium forces for freebodies
C        OUT10   10   output reactions at the supports
C        OUT11   11   output nodal coordinates
C        OUT12   12   output element connectivity
C        OUT28   28   output concrete/steel integration point data only
C
C ---- The following are CAEDS universal datasets
C
C        OUT20   20   output second Piola Kirchhoff stresses at the
C                     element nodes
C                21   output Cauchy stresses at the element nodes
C                22   output total element strains at the element nodes
C                23   output element elastic strains at the element node
C                24   output plastic element strains at the element node
C        OUT21   25   output nodal displacements
C        OUT22   26   output support reactions
C        OUT23   27   output nodal equilibrium loads
C
C
      READ (BUFF, *, END=180) IOINTR
  100 CONTINUE
      ICOMM = 0
      ISTART = 0
      IEND = 0
      INTR = 1
      IFG = 0
      IF (N .NE. 0) GO TO 120
  110 CONTINUE
      READ (LDEV11, '(A80)', END=160) BUFFER
  120 CONTINUE
      CALL COMPRO (N, NVAR)
C
      IOFLAG = IBSET(IOFLAG,20)
```

```
      IOFLAG = IBSET(IOFLAG,22)
      IF (COMM.EQ.'GEOM' .OR. COMM.EQ.'GEOMET') THEN
          IOFLAG = IBSET(IOFLAG,11)
          IOFLAG = IBSET(IOFLAG,12)
      ELSE IF (COMM .EQ. 'NSTRS') THEN
          IOFLAG = IBSET(IOFLAG,3)
          IOFLAG = IBSET(IOFLAG,21)
      ELSE IF (COMM .EQ. 'ISTRS') THEN
          IOFLAG = IBSET(IOFLAG,1)
      ELSE IF (COMM .EQ. 'NSTRN') THEN
          IOFLAG = IBSET(IOFLAG,4)
          IOFLAG = IBSET(IOFLAG,23)
          IOFLAG = IBSET(IOFLAG,24)
      ELSE IF (COMM .EQ. 'ISTRN') THEN
          IOFLAG = IBSET(IOFLAG,2)
      ELSE IF (COMM.EQ.'DISP' .OR. COMM.EQ.'DISPLA') THEN
          IOFLAG = IBSET(IOFLAG,7)
          IOFLAG = IBSET(IOFLAG,25)
      ELSE IF (COMM .EQ. 'REACTI') THEN
          IOFLAG = IBSET(IOFLAG,10)
          IOFLAG = IBSET(IOFLAG,26)
      ELSE IF (COMM .EQ. 'EQUILI') THEN
          IOFLAG = IBSET(IOFLAG,27)
          IOFLAG = IBSET(IOFLAG,8)
          IOFLAG = IBSET(IOFLAG,9)
      ELSE IF (COMM .EQ. 'CONCRE') THEN
          IOFLAG = IBSET(IOFLAG,28)
      ELSE IF (COMM.EQ.'ELEM' .OR. COMM.EQ.'ELEMEN') THEN
          READ (BUFF, *, END=180) ISTART
          IEND = ISTART
          INTR = 1
          IFG = 1
      ELSE IF (COMM .EQ. 'TO') THEN
          READ (BUFF, *, END=180) IEND
      ELSE IF (COMM .EQ. 'BY') THEN
          READ (BUFF, *, END=180) INTR
      ELSE IF (COMM.EQ.'NODE' .OR. COMM.EQ.'NODES') THEN
          READ (BUFF, *, END=180) ISTART
          IEND = ISTART
          INTR = 1
          IFG = 2
      ELSE IF (COMM .EQ. 'DONE') THEN
          GO TO 130
      ELSE IF (COMM .EQ. 'COMMEN') THEN
          GO TO 110
      ELSE IF (COMM .EQ. 'END') THEN
          RETURN
      ELSE
          ICOMM = 1
          GO TO 170
      ENDIF
C
      IF (N .NE. 0) THEN
          GO TO 120
      ELSE
          GO TO 110
      ENDIF
C
  130 CONTINUE
      IF (IFG .EQ. 1) THEN
          DO KK = ISTART, IEND, INTR
              IOEL(KK) = 1
          END DO
```

```
         GO TO 100
      ELSE IF (IFG .EQ. 2) THEN
         DO KK = ISTART, IEND, INTR
            IONOD(KK) = 1
         END DO
         GO TO 100
      ENDIF
C
  160 CONTINUE
      ICOMM = 2
  170 CONTINUE
      RETURN
  180 CONTINUE
      CALL ERRORS (3, 1, 'IOOUT ')
      END
C
C ======================= E B O R A S ================================
C
      SUBROUTINE EBORAS
C
C ====================================================================
C I
C I   P R O G R A M:
C I
C I   EBORAS (EBCDIC or ASCII) determines the integer value for
C I   the control characters used by the command processor COMPRO.
C I   This program also determines whether the machine used is an
C I   ASCII or EBCDIC machine.
C I
C ===================================================================
C
      IMPLICIT INTEGER (A-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C************************************************************
      COMMON/COMP1/SPACE,COMMA,ZERO,NINE,DOT,MINUS,EXCL,A,Z,QUOTE,EQUAL,
     1             PLUS,IUPER,K,E,D
C
      A = ICHAR('a')
      AU = ICHAR('A')
C
C ---- A = integer equivalent of the lower case A
C ---- AU= integer equivalent of the uper case A
C
C ---- IUPER is a number that must be added to the integer equivalent
C        of the lower case characters in order to convert to uper case.
C        For all machines (ASCII or EBCDIC) the increment between the
C        integer equivalents of the lower case and upper case characters
C        is fixed, hence  IUPER = AU - A
C
      IUPER = AU - A
C
      Z = ICHAR('z')
      SPACE = ICHAR(' ')
      COMMA = ICHAR(',')
      ZERO = ICHAR('0')
      NINE = ICHAR('9')
```

```
      DOT = ICHAR('.')
      MINUS = ICHAR('-')
      PLUS = ICHAR('+')
      EXCL = ICHAR('!')
      QUOTE = ICHAR('''''')
      EQUAL = ICHAR('=')
      E = ICHAR('e')
      D = ICHAR('d')
      K = 0
C
      RETURN
      END
C
C ====================== C O M P R O ===============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE COMPRO(NEXT,NVAR)
C
C ==================================================================
C I
C I   P R O G R A M:
C I
C I   COMPRO (Command Processor) is used extract the command which
C I   are recognizable by NLRC3D I/O module from each record (BUFFER)
C I   of the input file. The command and the parametres associated
C I   with it are the returned to the calling program.
C I
C I
C I   O N   E N T R Y:
C I
C I   BUFFER = chracter string 80 bytes long which contains one
C I            record of the input file (passsed by COMMON).
C I
C I
C I   O N   R E T U R N:
C I
C I   BUFF   = CHRACTER STRING 80 BYTES LONG WHICH CONTAINS THE
C I            PARAMETERS ASSOCIATED WITH THE COMMAND
C I
C I   COMM   = CHARACTER STRING 4 BYTES LONG WHICH CONTAINES THE
C I            FIRST FOUR CHARACTERS OF THE COMMAND.
C I
C I   NEXT   = is an integer which tells the calling routine whether
C I            the input file record BUFFER contains additional
C I            commands or not. When additional commands are detected
C I            the calling routine may repeat the call to
C I            COMPRO with the same BUFFER in order to extract the
C I            additional commands.
C I            =0; no additional commands detected
C I            =1; additional commands detected
C I
C I   NVAR   = the number of parameters associated with the extracted
C I            command COMM. These parameters are passed to the
C I            calling routine via the buffer BUFF in common COMP2
C I
C I
C ==========================================================
C
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
```

```
C
C ERRORS
C
C*************************************************************
      INTEGER SPACE,COMMA,ZERO,NINE,DOT,PLUS,EXCL,A,Z,QUOTE,EQUAL
      INTEGER E,D
      CHARACTER*80 BUFFER,BUFF
      CHARACTER*6 COMM
      CHARACTER*1 CCOMM
      COMMON/COMP1/SPACE,COMMA,ZERO,NINE,DOT,MINUS,EXCL,A,Z,QUOTE,EQUAL,
     1             PLUS,IUPER,K,E,D
      COMMON/COMP2/COMM,BUFFER,BUFF
C
C
C ---- INITIALIZE THE COMMAND comm AND THE BUFFER BUFF
C
      COMM = '      '
      BUFF = ' '
      NVAR = 0
C
C ---- DEFINITION OF LOCAL PARAMETERS.
C
C       IQUOTE = 0; NO QUOTES (') HAVE BEEN DETECTED
C                1; A SINGLE QUOTE HAS BEEN DETECTED
C
C       ICOMM    IS THE INTEGER EQUIVALENT OF THE UPPER CASE CHARACTERS
C                iT IS PRIMARILY USED WHEN CONVERTING LOWER CASE
C                CHARACTERS TO UPPER CASE CHARACTERS
C
C       ICOUNT   IS THE COUNTER USED TO KEEP COUNT OF THE NUMBER OF
C                CHARACTERS EXTRACTED FOR THE COMMAND COMM. If ICOUNT
C                IS GREATER THAN 6 THEN NO MORE CHARACTERS NEED TO
C                BE EXTRACTED.
C
C       INUM   = 0; NO NUMERIC DATA IS CURRENTLY BEING PROCESSED
C                1; NUMERIC DATA IS BEEING PROCESSED
C
C       ISPACE = 0; THE COMMAND IS BEEING EXTRACTED
C                1; A SPACE SEPERATOR HAS BEEN DETECTED AS THE END OF
C                   COMMAND.


C
C       ICHTR  = 0; NO CHARACTERS OTHER THAN SPACE HAVE BEEN DETECTED
C                1; AT LEAST ONE ALPHABETIC CHARACTER ASSOCIATED WITH
C                   A COMMAND HAS BEEN DETECTED.
C
      IQUOTE = 0
      ICOUNT = 0
      INUM = 0
      ISPACE = 0
      ICHTR = 0
      NEXT = 0
C
C ---- K1 IS THE POINTER FOR BUFF
C       K  IS THE POINTER FOR BUFFER
C
      K1 = 0
  100 CONTINUE
      K = K + 1
C
C ---- NCHAR IS THE INTEGER EQUIVALENT OF THE CHARACTER TO BE PROC.
C
      NCHAR = ICHAR(BUFFER(K:K))
```

```
      IF (NCHAR.EQ.EQUAL .OR. NCHAR.EQ.COMMA) NCHAR = SPACE
C
      IF (K .EQ. 80) THEN
          IF (ICHTR .NE. 1) COMM = 'COMMEN'
          GO TO 110
      ELSE IF (NCHAR .EQ. EXCL) THEN
          NEXT = 0
          IF (ICHTR .NE. 1) COMM = 'COMMEN'
          GO TO 110
      ELSE IF (NCHAR.EQ.SPACE .AND. INUM.EQ.1) THEN
          INUM = 0
          NVAR = NVAR + 1
          K1 = K1 + 1
          BUFF(K1:K1) = ' '
      ELSE IF (NCHAR.EQ.SPACE .AND. ICHTR.EQ.1) THEN
          ISPACE = 1
          K1 = K1 + 1
          BUFF(K1:K1) = ' '
      ELSE IF (NCHAR.EQ.QUOTE .OR. IQUOTE.EQ.1) THEN
          IF (NCHAR .EQ. QUOTE) THEN
              IF (IQUOTE .EQ. 1) THEN
                  IQUOTE = 0
                  NVAR = NVAR + 1
              ELSE IF (IQUOTE .EQ. 0) THEN
                  IQUOTE = 1
              ENDIF
          ENDIF
C
C ---- CONVERT LOWER CASE CHARACTERS TO UPPER CASE
C ---- ICOMM IS THE INTEGER EQUIVALENT OF THE UPPER CASE CHARACTER
C
          K1 = K1 + 1
          IF (NCHAR.GE.A .AND. NCHAR.LE.Z) THEN
              ICOMM = NCHAR + IUPER
          ELSE
              ICOMM = NCHAR
          ENDIF
          CCOMM = CHAR(ICOMM)
          BUFF(K1:K1) = CCOMM
      ELSE IF (NCHAR.EQ.PLUS .OR. NCHAR.EQ.MINUS .OR. NCHAR.EQ.DOT) THEN
          IF (ISPACE .EQ. 1) THEN
              INUM = 1
              K1 = K1 + 1
              BUFF(K1:K1) = BUFFER(K:K)
          ELSE
              ICOUNT = ICOUNT + 1
              IF (ICOUNT .LE. 6) COMM(ICOUNT:ICOUNT) = BUFFER(K:K)
          ENDIF
      ELSE IF (NCHAR.GE.ZERO .AND. NCHAR.LE.NINE) THEN
          IF (ISPACE .EQ. 1) THEN
              INUM = 1
              K1 = K1 + 1
              BUFF(K1:K1) = BUFFER(K:K)
          ELSE
              ICOUNT = ICOUNT + 1
              IF (ICOUNT .LE. 6) COMM(ICOUNT:ICOUNT) = BUFFER(K:K)
          ENDIF
      ELSE IF (ISPACE.EQ.1 .AND. NCHAR.NE.SPACE) THEN
          IF (INUM .EQ. 1) THEN
              IF (NCHAR.EQ.E .OR. NCHAR.EQ.D) THEN
                  ICOMM = NCHAR + IUPER
              ELSE
                  ICOMM = NCHAR
```

```
            ENDIF
            CCOMM = CHAR(ICOMM)
            IF (CCOMM.EQ.'E' .OR. CCOMM.EQ.'D') THEN
                K1 = K1 + 1
                BUFF(K1:K1) = CCOMM
            ELSE
                CALL ERRORS (1, 1, 'COMPRO')
            ENDIF
        ELSE
            NEXT = 1
            K = K - 1
            GO TO 110
        ENDIF
    ELSE IF (NCHAR .NE. SPACE) THEN
        ICHTR = 1
        ICOUNT = ICOUNT + 1
C
C ---- CONVERT LOWER CASE CHARACTERS TO UPPER CASE
C ---- ICOMM IS THE INTEGER EQUIVALENT OF THE UPPER CASE CHARACTER
C
        IF (NCHAR.GE.A .AND. NCHAR.LE.Z) THEN
            ICOMM = NCHAR + IUPER
        ELSE
            ICOMM = NCHAR
        ENDIF
        CCOMM = CHAR(ICOMM)
        IF (ICOUNT .LE. 6) COMM(ICOUNT:ICOUNT) = CCOMM
    ENDIF
    GO TO 100
C
  110 CONTINUE
    IF (NEXT .EQ. 0) K = 0
    RETURN
    END
C
C
C ==================================================================
C     INCLUDE (PROCESS)
      SUBROUTINE INANDY(PROPER,IOUT,IERROR,MATNUM)
C
C ==================================================================
C I
C I   P R O G R A M:
C I
C I   INANDY PROCESSES  INPUT ROUTINE FOR nlrc3d. tHIS PROGRAM READS
C I   EACH RECORD OF THE INPUT FILE AND EXTRACTS THE COMMAND BY
C I   CALLING SUBROUTINE COMPRO. THE EXTRACTED COMMANDS ARE THEN
C I   PROCESSED APPROPRIATELY.
C I
C I
C I   O N   E N T R Y:
C I
C I   IOUT   = output device number
C I
C I
C ==================================================================
C      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C**********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
```

```
C COMPRO    ERRORS
C
C************************************************************
      REAL*8 LTHICK
      CHARACTER*80 BUFFER,BUFF
      CHARACTER*6  COMM
      COMMON/LAYERA/ELLYFO(320,5000)
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1              NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC1(10),LDCONT,INCPTR
      COMMON/INPUTA/INFOEL(6,5000)
      COMMON/COMP2/COMM,BUFFER,BUFF
      COMMON/IALGTM/IFLAG5
      DIMENSION ELYMAT(320),PROPER(25,10)
      LDEV11 = 11
C
C.... ALL THE DIFFERENT MATERIALS ARE READ AS A GROUP ONETIME
C     AND MATERIAL NUMBERS ASSIGNED TO THEM SEQUENTIALLY IRESPECTIVE
C     OF AN ELEMENT/ELEMENT GROUP REQUIRING ANY PARTICULAR ELEMENT.
C
      MATNUM = 0
C
C..... FOR EACH ELEMENT OR GROUP OF ELEMENTS....
C
C     THE NUMBER OF LAYERS(DEFAULT ONE) IN THE ELEMENT(S) AND
C
C... THE FOLLOWING MUST BE SPECIFIED FOR EACH LAYER(NO DEFAULT)
C
C     LAYER THICKNESS ONE CONSTANT VALUE OR 4 CORNER VALUES
C     DISTANCE Z FROM ELEMENT NODE ONE CONSTANT VALUE OR 4 CORNER VALUES
C     INTEGRATION POINTS IN THREE LOCAL DIRECTIONS
C     DIRECTION COSINES FOR MATERIAL ORIENTATION(FOR STEEL LAYERS MUST)
C     MATERIAL PROPERTY NUMBER FOR GIVEN LAYER
C
  100 CONTINUE
      LTHICK = 0.
      ZI = 0.
      LAYNUM = 0
      N = 0
      ISTART = 0
      IEND = 0
      INTR = 1
C
      DO 110 IK = 1, 320
          ELYMAT(IK) = 0.0
  110 CONTINUE
C
      IF (O .NE. 0) GO TO 130
  120 CONTINUE
      READ (LDEV11, '(A80)', END=140) BUFFER
C
C ----  EXTRACT THE FIRST SIX CHARACTERS OF EACH COMMAND IN THE BUFFER
C        AND PLACE ALL VARIABLES ASSOCIATED WITH EACH COMMAND IN THE
C        INTERNAL FILE BUFF.
C
  130 CONTINUE
      CALL COMPRO (N, NVAR)
C
      IF (COMM.EQ.'ELEMEN' .OR. COMM.EQ.'ELEM') THEN
          READ (BUFF, *, END=180) ISTART
          IEND = ISTART
          INTR = 1
      ELSE IF (COMM .EQ. 'TO') THEN
```

```
          READ (BUFF, *, END=180) IEND
      ELSE IF (COMM .EQ. 'BY') THEN
          READ (BUFF, *, END=180) INTR
C
      ELSE IF (COMM .EQ. 'SECANT') THEN
          IFLAG5 = 0
      ELSE IF (COMM .EQ. 'TANGEN') THEN
          IFLAG5 = 1
C
      ELSE IF (COMM.EQ.'MATERI' .OR. COMM.EQ.'MAT') THEN
          MATNUM = MATNUM + 1
          IF (MATNUM .GT. 10) CALL ERRORS (3, 1, 'IANDMR')
      ELSE IF (COMM.EQ.'CONCRE' .OR. COMM.EQ.'CONC') THEN
          PROPER(25,MATNUM) = 1
      ELSE IF (COMM.EQ.'STEEL' .OR. COMM.EQ.'REINFO') THEN
          PROPER(25,MATNUM) = 0
      ELSE IF (COMM.EQ.'ELASTI' .OR. COMM.EQ.'ISOTRO') THEN
          PROPER(25,MATNUM) = 2
      ELSE IF (COMM.EQ.'E' .OR. COMM.EQ.'STIFF') THEN
          READ (BUFF, *, END=180) PROPER(1,MATNUM)
      ELSE IF (COMM.EQ.'NU' .OR. COMM.EQ.'POISSO') THEN
          READ (BUFF, *, END=180) PROPER(2,MATNUM)
      ELSE IF (COMM.EQ.'ATEMP' .OR. COMM.EQ.'THERMA') THEN
          READ (BUFF, *, END=180) PROPER(3,MATNUM)
      ELSE IF (COMM.EQ.'PHO' .OR. COMM.EQ.'WEIGHT') THEN
          READ (BUFF, *, END=180) PROPER(4,MATNUM)
      ELSE IF (COMM.EQ.'FC' .OR. COMM.EQ.'COMP') THEN
          READ (BUFF, *, END=180) PROPER(5,MATNUM)
      ELSE IF (COMM.EQ.'FT' .OR. COMM.EQ.'TENS') THEN
          READ (BUFF, *, END=180) PROPER(6,MATNUM)
      ELSE IF (COMM .EQ. 'EPSC') THEN
          READ (BUFF, *, END=180) PROPER(7,MATNUM)
      ELSE IF (COMM .EQ. 'EPSU') THEN
          READ (BUFF, *, END=180) PROPER(8,MATNUM)
      ELSE IF (COMM .EQ. 'DSOFT') THEN
          READ (BUFF, *, END=180) PROPER(9,MATNUM)
      ELSE IF (COMM.EQ.'BSHR' .OR. COMM.EQ.'SHEAR') THEN
          READ (BUFF, *, END=180) PROPER(10,MATNUM)
      ELSE IF (COMM.EQ.'GF' .OR. COMM.EQ.'ENERGY') THEN
          READ (BUFF, *, END=180) PROPER(11,MATNUM)
      ELSE IF (COMM.EQ.'NUF' .OR. COMM.EQ.'FPOISS') THEN
          READ (BUFF, *, END=180) PROPER(12,MATNUM)
      ELSE IF (COMM .EQ. 'BETAA') THEN
          READ (BUFF, *, END=180) PROPER(13,MATNUM)
      ELSE IF (COMM .EQ. 'FCSIG') THEN
          READ (BUFF, *, END=180) PROPER(14,MATNUM)
      ELSE IF (COMM.EQ.'ATHETA' .OR. COMM.EQ.'THRESH') THEN
          READ (BUFF, *, END=180) PROPER(15,MATNUM)
      ELSE IF (COMM .EQ. 'TSFLAG') THEN
          READ (BUFF, *, END=180) PROPER(16,MATNUM)
      ELSE IF (COMM .EQ. 'TSTERM') THEN
          READ (BUFF, *, END=180) PROPER(17,MATNUM)
      ELSE IF (COMM .EQ. 'TSTIFG') THEN
          READ (BUFF, *, END=180) PROPER(24,MATNUM)
      ELSE IF (COMM .EQ. 'CSOFTN') THEN
          READ (BUFF, *, END=180) PROPER(23,MATNUM)
      ELSE IF (COMM .EQ. 'YEILD') THEN
          READ (BUFF, *, END=180) PROPER(5,MATNUM)
      ELSE IF (COMM .EQ. 'AHARD') THEN
          READ (BUFF, *, END=180) PROPER(6,MATNUM)
      ELSE IF (COMM .EQ. 'EPSBRK') THEN
          READ (BUFF, *, END=180) PROPER(7,MATNUM)
      ELSE IF (COMM.EQ.'TOLERE' .OR. COMM.EQ.'TOLER') THEN
```

```
            READ (BUFF, *, END=180) PROPER(22,MATNUM)
C
         ELSE IF (COMM.EQ.'INTLYR' .OR. COMM.EQ.'ILAYER') THEN
            READ (BUFF, *, END=180) ELYMAT(320)
            LAYNUM = 0
         ELSE IF (COMM .EQ. 'LMAT') THEN
            LAYNUM = LAYNUM + 1
            IF (LAYNUM .GT. 15) CALL ERRORS (47, 1, 'ELEMEN')
            INDEX = 21*(LAYNUM-1) + 1
            READ (BUFF, *, END=180) ELYMAT(INDEX)
         ELSE IF (COMM .EQ. 'ORIENT') THEN
            INDEX = 21*(LAYNUM-1) + 10
            READ (BUFF, *, END=180) ((ELYMAT(ICOUNT)), ICOUNT = INDEX,
     1          INDEX + 8)
         ELSE IF (COMM .EQ. 'LTHICK') THEN
            READ (BUFF, *, END=180) LTHICK
            INDEX = 21*(LAYNUM-1) + 2
            ELYMAT(INDEX) = LTHICK
            ELYMAT(INDEX+1) = LTHICK
            ELYMAT(INDEX+2) = LTHICK
            ELYMAT(INDEX+3) = LTHICK
         ELSE IF (COMM .EQ. 'TL1') THEN
            INDEX = 21*(LAYNUM-1) + 2
            READ (BUFF, *, END=180) ELYMAT(INDEX)
         ELSE IF (COMM .EQ. 'TL2') THEN
            INDEX = 21*(LAYNUM-1) + 3
            READ (BUFF, *, END=180) ELYMAT(INDEX)
         ELSE IF (COMM .EQ. 'TL3') THEN
            INDEX = 21*(LAYNUM-1) + 4
            READ (BUFF, *, END=180) ELYMAT(INDEX)
         ELSE IF (COMM .EQ. 'TL4') THEN
            INDEX = 21*(LAYNUM-1) + 5
            READ (BUFF, *, END=180) ELYMAT(INDEX)
         ELSE IF (COMM .EQ. 'ZI') THEN
            READ (BUFF, *, END=180) ZI
            INDEX = 21*(LAYNUM-1) + 6
            ELYMAT(INDEX) = ZI
            ELYMAT(INDEX+1) = ZI
            ELYMAT(INDEX+2) = ZI
            ELYMAT(INDEX+3) = ZI
         ELSE IF (COMM .EQ. 'Z1') THEN
            INDEX = 21*(LAYNUM-1) + 6
            READ (BUFF, *, END=180) ELYMAT(INDEX)
         ELSE IF (COMM .EQ. 'Z2') THEN
            INDEX = 21*(LAYNUM-1) + 7
            READ (BUFF, *, END=180) ELYMAT(INDEX)
         ELSE IF (COMM .EQ. 'Z3') THEN
            INDEX = 21*(LAYNUM-1) + 8
            READ (BUFF, *, END=180) ELYMAT(INDEX)
         ELSE IF (COMM .EQ. 'Z4') THEN
            INDEX = 21*(LAYNUM-1) + 9
            READ (BUFF, *, END=180) ELYMAT(INDEX)
         ELSE IF (COMM .EQ. 'NIPXI') THEN
            INDEX = 21*(LAYNUM-1) + 19
            READ (BUFF, *, END=180) ELYMAT(INDEX)
         ELSE IF (COMM .EQ. 'NIPETA') THEN
            INDEX = 21*(LAYNUM-1) + 20
            READ (BUFF, *, END=180) ELYMAT(INDEX)
         ELSE IF (COMM .EQ. 'NIPSI') THEN
            INDEX = 21*(LAYNUM-1) + 21
            READ (BUFF, *, END=180) ELYMAT(INDEX)
         ELSE IF (COMM .EQ. 'END') THEN
            GO TO 190
```

```
      ELSE IF (COMM .EQ. 'DONE') THEN
          ICOMM = 1
          GO TO 150
      ENDIF
C
      IF (N .NE. 0) THEN
          GO TO 130
      ELSE
          GO TO 120
      ENDIF
C
  140 CONTINUE
      ICOMM = 2
  150 CONTINUE
      DO 170 K = ISTART, IEND, INTR
C#################
          ELLYFO(320,K) = ELYMAT(320)
          NLAYRS = INT(ELYMAT(320))
          IF (NLAYRS .EQ. 0) THEN
              NLAYRS = 1
              ELLYFO(320,K) = 1
          ENDIF
          DO 160 ILY = 1, NLAYRS
              ICNT = 21*(ILY-1) + 1
              ELLYFO(ICNT,K) = ELYMAT(ICNT)
              ELLYFO(ICNT+1,K) = ELYMAT(ICNT+1)
              ELLYFO(ICNT+2,K) = ELYMAT(ICNT+2)
              ELLYFO(ICNT+3,K) = ELYMAT(ICNT+3)
              ELLYFO(ICNT+4,K) = ELYMAT(ICNT+4)
              ELLYFO(ICNT+5,K) = ELYMAT(ICNT+5)
              ELLYFO(ICNT+6,K) = ELYMAT(ICNT+6)
              ELLYFO(ICNT+7,K) = ELYMAT(ICNT+7)
              ELLYFO(ICNT+8,K) = ELYMAT(ICNT+8)
              ELLYFO(ICNT+9,K) = ELYMAT(ICNT+9)
              ELLYFO(ICNT+10,K) = ELYMAT(ICNT+10)
              ELLYFO(ICNT+11,K) = ELYMAT(ICNT+11)
              ELLYFO(ICNT+12,K) = ELYMAT(ICNT+12)
              ELLYFO(ICNT+13,K) = ELYMAT(ICNT+13)
              ELLYFO(ICNT+14,K) = ELYMAT(ICNT+14)
              ELLYFO(ICNT+15,K) = ELYMAT(ICNT+15)
              ELLYFO(ICNT+16,K) = ELYMAT(ICNT+16)
              ELLYFO(ICNT+17,K) = ELYMAT(ICNT+17)
              ELLYFO(ICNT+18,K) = ELYMAT(ICNT+18)
              ELLYFO(ICNT+19,K) = ELYMAT(ICNT+19)
              ELLYFO(ICNT+20,K) = ELYMAT(ICNT+20)
  160     CONTINUE
  170 CONTINUE
      IF (ICOMM .EQ. 2) GO TO 180
      GO TO 100
  180 CONTINUE
      CALL ERRORS (3, 1, 'INANDY')
  190 CONTINUE
      RETURN
      END
C ==================================================================
C     INCLUDE (PROCESS)
      SUBROUTINE IOHIST(IOUT,IERROR)
C
C =================================================================
C I
C I   P R O G R A M:
C I
C I   IOHIST PROCESSES  INPUT ROUTINE FOR NLRC3D. tHIS PROGRAM READS
```

```
C I   EACH RECORD OF THE INPUT FILE AND EXTRACTS THE COMMAND BY
C I   CALLING SUBROUTINE compro. tHE EXTRACTED COMMANDS ARE THEN
C I   PROCESSED APPROPRIATELY.
C I
C I
C I   O N   E N T R Y:
C I
C I   IOUT    = output device number
C I
C I
C ===================================================================
C       IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C COMPRO    ERRORS
C
C*************************************************************
      CHARACTER*80 BUFFER,BUFF
      CHARACTER*6  COMM
      COMMON/COMP2/COMM,BUFFER,BUFF
      COMMON/PATH/HISTX(10,30),HISTY(10,30),IPATH(10)
C
C ---- READ ONE LINE OF THE INPUT FILE AND STORE IN BUFFER
C
      LDEV11 = 11
      N = 0
      ICOUNT = 0
      HNUM = 3
C
      IF (0 .NE. 0) GO TO 110
  100 CONTINUE
      READ (LDEV11, '(A80)', END=130) BUFFER
C
C ----  EXTRACT THE FIRST SIX CHARACTERS OF EACH COMMAND IN THE BUFFER
C        AND PLACE ALL VARIABLES ASSOCIATED WITH EACH COMMAND IN THE
C        INTERNAL FILE BUFF.
C
  110 CONTINUE
      CALL COMPRO (N, NVAR)
c
      IF (COMM .EQ. 'POINTS') THEN
          HNUM = HNUM + 1
          ICOUNT = 0
          READ (BUFF, *, END=120) IPATH(HNUM)
      ELSE IF (COMM .EQ. 'HXAXIS') THEN
          ICOUNT = ICOUNT + 1
          READ (BUFF, *, END=120) HISTX(HNUM,ICOUNT)
      ELSE IF (COMM .EQ. 'HYAXIS') THEN
          IF (HISTY(HNUM,ICOUNT) .NE. 0.0) CALL ERRORS (4, 1, 'IOHIST')
          READ (BUFF, *, END=120) HISTY(HNUM,ICOUNT)
      ELSE IF (COMM .EQ. 'END') THEN
          RETURN
      ELSE
          GO TO 140
      ENDIF
      IF (N .EQ. 0) THEN
          GO TO 100
      ELSE
          GO TO 110
```

```
      ENDIF
  120 CONTINUE
      CALL ERRORS (4, 1, 'IOHIST')
  130 CONTINUE
  140 CONTINUE
      RETURN
      END
C ==================================================================
C     INCLUDE (PROCESS)
      SUBROUTINE IOCNST(LAMINA,IOUT,IERROR,ICOMM)
C
C I
C I   P R O G R A M:
C I
C I   IOCNST PROCESSES  INPUT ROUTINE FOR nlrc3d. tHIS PROGRAM READS
C I   EACH RECORD OF THE INPUT FILE AND EXTRACTS THE COMMAND BY
C I   CALLING SUBROUTINE COMPRO. tHE EXTRACTED COMMANDS ARE THEN
C I   PROCESSED APPROPRIATELY.
C I
C I
C I   O N   E N T R Y:
C I
C I   IOUT   = output device number
C I
C I
C ==================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C**********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C COMPRO    ERRORS
C
C**********************************************************
      CHARACTER*80 BUFFER,BUFF
      CHARACTER*6  COMM
      COMMON/COMP2/COMM,BUFFER,BUFF
      DIMENSION LAMINA(5000,2)
      LDEV11 = 11
C
C
C..... FOR EACH ELEMENT OR GROUP OF ELEMENTS....
C
C     THE NUMBER OF LAYERS(DEFAULT ONE) IN THE ELEMENT(S) AND
C
C
      N = 0
      ISTART = 0
      IEND = 0
      INTR = 1
      ICOUNT = 0
C
C
  100 CONTINUE
      IF (N .NE. 0) GO TO 120
  110 CONTINUE
      READ (LDEV11, '(A80)', END=180) BUFFER
C
C ----  EXTRACT THE FIRST SIX CHARACTERS OF EACH COMMAND IN THE BUFFER
C        AND PLACE ALL VARIABLES ASSOCIATED WITH EACH COMMAND IN THE
```

```
C       INTERNAL FILE BUFF.
C
  120 CONTINUE
      CALL COMPRO (N, NVAR)
C
      IF (COMM .EQ. 'STACK') THEN
          ISTART = 0
          IEND = 0
          INTR = 1
      ELSE IF (COMM.EQ.'ELEMEN' .OR. COMM.EQ.'ELEM') THEN
          READ (BUFF, *, END=160) ISTART
          IEND = ISTART
          INTR = 1
      ELSE IF (COMM .EQ. 'TO') THEN
          READ (BUFF, *, END=160) IEND
      ELSE IF (COMM .EQ. 'BY') THEN
          READ (BUFF, *, END=160) INTR
      ELSE IF (COMM .EQ. 'AND') THEN
          READ (BUFF, *, END=160) IEND
          ICOUNT = ICOUNT + 1
          LAMINA(ICOUNT,1) = ISTART
          LAMINA(ICOUNT,2) = IEND
      ELSE IF (COMM .EQ. 'END') THEN
          ICOMM = 0
          GO TO 170
      ELSE IF (COMM .EQ. 'STOP') THEN
          ICOMM = 1
          GO TO 130
      ELSE IF (COMM .EQ. 'DONE') THEN
          ICOMM = 1
          GO TO 100
      ENDIF
C
      IF (N .NE. 0) THEN
          GO TO 120
      ELSE
          GO TO 110
      ENDIF
C
  130 CONTINUE
      DO 150 K = ISTART, IEND, INTR
          IF (K .EQ. IEND) GO TO 140
          ICOUNT = ICOUNT + 1
          LAMINA(ICOUNT,1) = K
          LAMINA(ICOUNT,2) = K + INTR
  140     CONTINUE
  150 CONTINUE
      IF (ICOMM .EQ. 2) GO TO 160
      GO TO 100
  160 CONTINUE
      CALL ERRORS (3, 1, 'IOCNST')
  170 CONTINUE
      RETURN
  180 CONTINUE
      ICOMM = 2
      END
C**************************************************************
C     INCLUDE(PROCESS)
      SUBROUTINE CNPROP(LAMINA,IOUT,IERROR)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*************************************************************
```

```
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ELINFO    ELINTM    WCONST
C
C************************************************************
      REAL*4 XYZ,THICK1,THICK2
      COMMON/MPCS/COEFMP(40000),ALAMB(40000),MPCDOF(40000),
     $           MPCADR(2,5000),NMPC,MPCPNT,MAXMPC
C
      COMMON/LAYERA/ELLYFO(320,5000)
      COMMON/SAFE2/R(60000),IDOF(60000),JDIAG(60000)
      COMMON/INPUT2/NOP(20,5000)
      COMMON/INPUT3/XYZ(3,10000)
      COMMON/CONS/ICHECK(60000)
      COMMON/LPROP/PROPER(25,15)
      DIMENSION LAMINA(5000,2),THICK1(9),THICK2(9)
C
      ILINK = 0
      NMPC = 0
      MPCPNT = 0
      DO KM = 1, 60000
          ICHECK(KM) = 0
      END DO
      DO 220 ICOUNT = 1, 5000
          IF (LAMINA(ICOUNT,1) .NE. 0) THEN
              IELEM1 = LAMINA(ICOUNT,1)
              IELEM2 = LAMINA(ICOUNT,2)
              CALL ELINFO (IELEM1, ITYPE1, NNEL1, IFLAG1, ISTART, LINES)
              CALL ELINTM (IELEM1, IDENT1, INTCOD, NIPXI, NIPETA, NIPSI
     1            , MATNUM, THICK1)
              CALL ELINFO (IELEM2, ITYPE2, NNEL2, IFLAG2, ISTART, LINES)
              CALL ELINTM (IELEM2, IDENT2, INTCOD, NIPXI, NIPETA, NIPSI
     1            , MATNUM, THICK2)
              IF (ITYPE1.EQ.ITYPE2 .AND. IFLAG1.EQ.IFLAG2) THEN
                  DO 210 ND1 = 1, NNEL1
                      NODE1 = NOP(ND1,IELEM1)
                      TH1 = THICK1(ND1)
                      X1 = XYZ(1,NODE1)
                      Y1 = XYZ(2,NODE1)
                      Z1 = XYZ(3,NODE1)
                      DIST = 1.E30
                      DIST1 = 1.E30
                      NC = 0
                      TH2 = 0.0
                      DO 110 ND2 = 1, NNEL2
                          NODE2 = NOP(ND2,IELEM2)
                          X2 = XYZ(1,NODE2)
                          Y2 = XYZ(2,NODE2)
                          Z2 = XYZ(3,NODE2)
C
                          DIST = DSQRT((X1-X2)**2+(Y1-Y2)**2+(Z1-Z2)**2)
                          IF (DIST .LE. DIST1) THEN
                              NC = NODE2
                              TH2 = THICK2(ND2)
                              DIST1 = DIST
                          ENDIF
 110                  CONTINUE
                      NODE2 = NC
C
                      IDENT1 = (NODE1-1)*6 + 3
                      IDENT2 = (NODE2-1)*6 + 3
                      IF (ICHECK(IDENT1).EQ.1 .AND. ICHECK(IDENT2).EQ.1
```

```
     1                ) GO TO 120
                    ICHECK(IDENT1) = 1
                    ICHECK(IDENT2) = 1
C
                    IDENT1 = (NODE1-1)*6 + 3
                    IDENT2 = (NODE2-1)*6 + 3
                    IF (IDOF(IDENT1).GT.0 .OR. IDOF(IDENT2).GT.0) THEN
                        IDOF(IDENT1) = 1
                        IDOF(IDENT2) = 1
                        GO TO 120
                    ENDIF
                    ILINK = ILINK + 1
                    NMPC = NMPC + 1
                    MPCPNT = MPCPNT + 1
                    MPCDOF(MPCPNT) = 0
                    COEFMP(MPCPNT) = 0.0
                    MPCADR(1,NMPC) = MPCPNT
                    MPCADR(2,NMPC) = 3
                    MPCPNT = MPCPNT + 1
                    MPCDOF(MPCPNT) = IDENT1
                    COEFMP(MPCPNT) = 1.0
                    MPCPNT = MPCPNT + 1
                    MPCDOF(MPCPNT) = IDENT2
                    COEFMP(MPCPNT) = -1.0
C
  120               CONTINUE
                    IDENT1 = (NODE1-1)*6 + 1
                    IDENT2 = (NODE1-1)*6 + 5
                    IDENT3 = (NODE2-1)*6 + 1
                    IDENT4 = (NODE2-1)*6 + 5
c
                    IF (ICHECK(IDENT1).EQ.1 .AND. ICHECK(IDENT2).EQ.1
     1                   .AND. ICHECK(IDENT3).EQ.1 .AND. ICHECK(IDENT4
     2                ).EQ.1) GO TO 160
                    ICHECK(IDENT1) = 1
                    ICHECK(IDENT2) = 1
                    ICHECK(IDENT3) = 1
                    ICHECK(IDENT4) = 1
C
                    IF (IDOF(IDENT1).GT.0 .AND. IDOF(IDENT2).GT.0
     1                   .AND. IDOF(IDENT3).GT.0 .AND. IDOF(IDENT4)
     2                .GT.0) GO TO 160
C
                    IREST = 0
                    IF (IDOF(IDENT1) .GT. 0) IREST = 1
                    IF (IDOF(IDENT2) .GT. 0) IREST = IREST + 1
                    IF (IDOF(IDENT3) .GT. 0) IREST = IREST + 1
                    IF (IDOF(IDENT4) .GT. 0) IREST = IREST + 1
C
                    ILINK = ILINK + 1
                    NMPC = NMPC + 1
                    MPCPNT = MPCPNT + 1
                    MPCDOF(MPCPNT) = 0
                    COEFMP(MPCPNT) = 0.0
                    MPCADR(1,NMPC) = MPCPNT
                    MPCADR(2,NMPC) = 5 - IREST
                    IF (IDOF(IDENT1) .GT. 0) GO TO 130
                    MPCPNT = MPCPNT + 1
                    MPCDOF(MPCPNT) = IDENT1
                    COEFMP(MPCPNT) = 1.0
  130               CONTINUE
                    IF (IDOF(IDENT2) .GT. 0) GO TO 140
                    MPCPNT = MPCPNT + 1
```

```
                  MPCDOF(MPCPNT) = IDENT2
                  COEFMP(MPCPNT) = -1.0*TH1/2.0
  140             CONTINUE
                  IF (IDOF(IDENT3) .GT. 0) GO TO 150
                  MPCPNT = MPCPNT + 1
                  MPCDOF(MPCPNT) = IDENT3
                  COEFMP(MPCPNT) = -1.0
  150             CONTINUE
                  IF (IDOF(IDENT4) .GT. 0) GO TO 160
                  MPCPNT = MPCPNT + 1
                  MPCDOF(MPCPNT) = IDENT4
                  COEFMP(MPCPNT) = -1.0*TH2/2.0
C
C           ICOUNT=ICOUNT+1
  160             CONTINUE
                  IDENT1 = (NODE1-1)*6 + 2
                  IDENT2 = (NODE1-1)*6 + 4
                  IDENT3 = (NODE2-1)*6 + 2
                  IDENT4 = (NODE2-1)*6 + 4
c
                  IF (ICHECK(IDENT1).EQ.1 .AND. ICHECK(IDENT2).EQ.1
     1                 .AND. ICHECK(IDENT3).EQ.1 .AND. ICHECK(IDENT4
     2                ).EQ.1) GO TO 200
                  ICHECK(IDENT1) = 1
                  ICHECK(IDENT2) = 1
                  ICHECK(IDENT3) = 1
                  ICHECK(IDENT4) = 1
C
                  IF (IDOF(IDENT1).GT.0 .AND. IDOF(IDENT2).GT.0
     1                 .AND. IDOF(IDENT3).GT.0 .AND. IDOF(IDENT4)
     2                .GT.0) GO TO 200
C
                  IREST = 0
                  IF (IDOF(IDENT1) .GT. 0) IREST = 1
                  IF (IDOF(IDENT2) .GT. 0) IREST = IREST + 1
                  IF (IDOF(IDENT3) .GT. 0) IREST = IREST + 1
                  IF (IDOF(IDENT4) .GT. 0) IREST = IREST + 1
c
                  ILINK = ILINK + 1
                  NMPC = NMPC + 1
                  MPCPNT = MPCPNT + 1
                  MPCDOF(MPCPNT) = 0
                  COEFMP(MPCPNT) = 0.0
                  MPCADR(1,NMPC) = MPCPNT
                  MPCADR(2,NMPC) = 5 - IREST
                  IF (IDOF(IDENT1) .GT. 0) GO TO 170
                  MPCPNT = MPCPNT + 1
                  MPCDOF(MPCPNT) = IDENT1
                  COEFMP(MPCPNT) = 1.0
  170             CONTINUE
                  IF (IDOF(IDENT2) .GT. 0) GO TO 180
                  MPCPNT = MPCPNT + 1
                  MPCDOF(MPCPNT) = IDENT2
                  COEFMP(MPCPNT) = 1.0*TH1/2.0
  180             CONTINUE
                  IF (IDOF(IDENT3) .GT. 0) GO TO 190
                  MPCPNT = MPCPNT + 1
                  MPCDOF(MPCPNT) = IDENT3
                  COEFMP(MPCPNT) = -1.0
  190             CONTINUE
                  IF (IDOF(IDENT4) .GT. 0) GO TO 200
                  MPCPNT = MPCPNT + 1
                  MPCDOF(MPCPNT) = IDENT4
```

```
                    COEFMP(MPCPNT) = 1.0*TH2/2.0
C
  200               CONTINUE
  210            CONTINUE
              ENDIF
           ENDIF
  220 CONTINUE
c
      CALL WCONST
c
      RETURN
      END
C ====================== S U L O A D ============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE SULOAD(IDIM,N,NNDF,ICOMM,LDEV11,IOUT)
C
C ================================================================
C I
C I   P R O G R A M:


C I
C I   SULOAD READS AND STORES THE SURFACE LOADS
C I
C I
C I   O N   E N T R Y:
C I
C I   IDIM   = PHYSICAL DIMENSION OF THE PROBLEM (I.E. 2D OR 3D)
C I   NNDF   = NUMBER OF NODAL DEGREES OF FREEDOM
C I   N      = 0; BUFFER CONTAINS NO ADDITIONAL COMMANDS
C I             1; bUFFER CONTAINES AT LEAST ONE ADDITIONAL COMMAND
C I   LDEV11 = UNIT NUMBER FOR THE INPUT FILE
C I   IOUT   = UNIT NUMBER FOR THE OUTPUT FILE
c i
C I
C I   O N   R E T U R N:
C I
C I   N      = 0; BUFFER CONTAINS NO ADDITIONAL COMMANDS
C I             1; bUFFER CONTAINES AT LEAST ONE ADDITIONAL COMMAND
C I
C I   ICOMM  = IDENTIFIES THE ACTION TO BE TAKEN WHEN CONTROL IS
C I             RETURNED TO THE CALLING ROUTINE .
C I             =0; READ THE NEXT INPUT LINE
C I             =1; tHE CURRENT COMMAND WAS NOT RESOLVED BY IODIS.
C I                cALLING ROUTINE SHOULD PROCESS THIS COMMAND
C I             =2; END OF FILE IS REACHED, DO NOT TRY TO READ ANY
C I                MORE LINES.
C I
C ================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C**************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C COMPRO    ERRORS
C
C**************************************************************
      CHARACTER*80 BUFFER,BUFF
      CHARACTER*6 COMM
      COMMON/COMP2/COMM,BUFFER,BUFF
```

```
      COMMON/SURLOD/PRES(5000,9),ISULOD(5000)
      DIMENSION P(9)
C
      ITIME = 0
      ICOMM = 0
      ISTART = 0
      IEND = 0
      INTR = 0
      ICHK = 0
      P11 = 0.0
      P22 = 0.0
      P33 = 0.0
      P44 = 0.0
      PRESUR = 0.0
C
      IF (N .NE. 0) GO TO 110
  100 CONTINUE
      READ (LDEV11, '(A80)', END=190) BUFFER
  110 CONTINUE
      CALL COMPRO (N, NVAR)
C
      IF (COMM.EQ.'ELEM' .OR. COMM.EQ.'ELEMEN') THEN
          IF (ITIME .EQ. 1) THEN
              ASSIGN 120 TO NEXT
          ENDIF
  120     CONTINUE
          READ (BUFF, *, END=200) ISTART
          ITIME = 1
          IEND = ISTART
          INTR = 1
C
C ---- IDCORD IS THE FLAG FOR THE TRANSLATION COORDINATE DEFINITION.
C             = 0; A LOCAL COORDINATE SYSTEM IS NOT DEFINED
C             = 1; A LOCAL COORDINATE SYSTEM IS DEFINED
C ---- IRCORD IS THE FLAG FOR THE ROTATIONAL COORDINATE DEFINITION.
C             = 0; A LOCAL COORDINATE SYSTEM IS NOT DEFINED
C             = 1; A LOCAL COORDINATE SYSTEM IS DEFINED
C
          DO 130 K1 = 1, 9
              P(K1) = 0.
  130     CONTINUE
      ELSE IF (COMM .EQ. 'TO') THEN
          READ (BUFF, *, END=200) IEND
      ELSE IF (COMM .EQ. 'BY') THEN
          READ (BUFF, *, END=200) INTR
      ELSE IF (COMM .EQ. 'PRESUR') THEN
          READ (BUFF, *, END=200) PRESUR
      ELSE IF (COMM .EQ. 'P1') THEN
          READ (BUFF, *, END=200) P11
      ELSE IF (COMM .EQ. 'P2') THEN
          READ (BUFF, *, END=200) P22
      ELSE IF (COMM .EQ. 'P3') THEN
          READ (BUFF, *, END=200) P33
      ELSE IF (COMM .EQ. 'P4') THEN
          READ (BUFF, *, END=200) P44
      ELSE IF (COMM .EQ. 'TOP') THEN
          ICHK = 1
      ELSE IF (COMM .EQ. 'BOTTOM') THEN
          ICHK = -1
      ELSE IF (COMM .EQ. 'END') THEN
          ICOMM = 0
          ASSIGN 140 TO NEXT
          GO TO 150
```

```
  140     CONTINUE
          RETURN
      ELSE IF (COMM .EQ. 'COMMEN') THEN
          GO TO 100
      ELSE
          ICOMM = 1
          ASSIGN 140 TO NEXT
          GO TO 150
      ENDIF
C
      IF (N .NE. 0) THEN
          GO TO 110
      ELSE
          GO TO 100
      ENDIF
C
C
  150 CONTINUE
      IF (PRESUR .NE. 0.0) THEN
          DO IM = 1, 9
              P(IM) = PRESUR
          END DO
          PRESUR = 0.0
      ENDIF
      IF(P11.NE.0.0.OR.P22.NE.0.0.OR.P33.NE.0.0.OR.P44.NE.0.0)THEN
          P(1) = P11
          P(2) = P22
          P(3) = P33
          P(4) = P44
          P(5) = (P11+P22)/2.0
          P(6) = (P22+P33)/2.0
          P(7) = (P33+P44)/2.0
          P(8) = (P44+P11)/2.0
          P(9) = (P11+P22+P33+P44)/4.0
          P11 = 0.0
          P22 = 0.0
          P33 = 0.0
          P44 = 0.0
      ENDIF
      DO 180 K1 = ISTART, IEND, INTR
          DO 170 IDIR = 1, 9
              ISULOD(K1) = ICHK
              IF (P(IDIR) .NE. 0.) PRES(K1,IDIR) = P(IDIR)
  170     CONTINUE
  180 CONTINUE
      ICHK = 0
C
      GO TO NEXT
C
  190 CONTINUE
      ICOMM = 2
      RETURN
  200 CONTINUE
      CALL ERRORS (3, 1, 'SULOAD')
      STOP
      END
C
C
C ======================= L O A D ================================
C
C     INCLUDE (PROCESS)
      SUBROUTINE LOAD (R,IERROR)
C
```

```
C ================================================================
C I
C I    P R O G R A M
C I
C I    LOAD ASSEMBLES THE LOAD VECTOR BY CONSIDERING THE
C I    EXTERNALY APPLIED LOADS AND THE GRAVITY LOADS WHICH ARE
C I    SUPPERIMPOSED ON THE STRUCTERE.
C I
C I
C I    A R G U M E N T   L I S T
C I
C I      R(I)      =  LOAD VECTOR TO BE ASSEMBLED
C I      IERROR    =  ERROR CODE =0; NO ERROR  0<; ERROR
C I
C I
C I    C O M M O N   B L O C K S
C I
C I    REFFER TO THE COMMON BLOCK DISCRIPTIONS.
C I      N(I,J)    =  SHAPE FUNCTION FOR NODE I AT INTEGR. POINT J
C I      W(I)      =  GAUSSIAN WEIGTHING FUNCTIONS
C I      XGAUSS    =  X COORDINATE OF THE GAUSSIAN POINTS IN THE ELEM.
C I      WGTX(I)   =  SPECIFIC WEIGTH OF MATERIAL I IN THE X DIR.
C I      WGTY(I)   =  SPECIFIC WEIGTH OF MATERIAL I IN THE Y DIR.
C I      WGTZ(I)   =  SPECIFIC WEIGTH OF MATERIAL I IN THE Z DIR.
C I      THICK     =  THICKNESS OF THE ELEMENTS FOR PLANE STR & STN
C I
C ================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ELINFO    ELINTM    LYINFO    ISH3DG    GAUSS
C ISHSHL    SHNORM    DIRVEC    JACB3D    GETTHK    JACSHL
C ISH2DG    JACB2D
C
C***********************************************************
      REAL*8 N,NXI,NETA,NSI,SI,LTHICK
      REAL*4 SHELLZ,THICK
      INTEGER ELNUM
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/INPUT2/NOP(20,5000)
      COMMON/SHLDIR/VECT(3,3,9)
      COMMON/INPUT6/WGTX(10),WGTY(10),WGTZ(10)
      COMMON/INPUT7/RX(60000),RY(60000),RZ(60000)
      COMMON/INPUT9/THICK(9),IFLAG
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1              NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC1(10),LDCONT,INCPTR
      COMMON/SKTR2/SHELLZ(3,10000)
      COMMON/INPUTE/ISPB(10000)
      COMMON/ISHAP1/N(20,27),NXI(20,27),NETA(20,27),NSI(20,27),SI(27)
      COMMON/ISHAP2/W(27)
      COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
      COMMON/TRANS/DC(3 , 3)
      COMMON/SURLOD/PRES(5000,9),ISULOD(5000)
      DIMENSION R( 1 ),DUMMY( 3 ),WXI(4),WETA(4),WSI(4),XI(4),ETA(4),
     $ SIZ(4),AMX(60000),AMY(60000),AMZ(60000)
C
C ----- FIND THE CONTRIBUTION OF THE GRAVITY WEIGTHS FO 2D ELEMENTS
```

```
C
      DO KL = 1, 60000
         AMX(KL) = 0.0
         AMY(KL) = 0.0
         AMZ(KL) = 0.0
      END DO
      DO 240 ELNUM = 1, NELEM
C
         CALL ELINFO (ELNUM, ITYPE, NNEL, IFLAG, ISTART, LINES)
         CALL ELINTM (ELNUM, IDENT, INTCOD, NIPXI, NIPETA, NIPSI,
     1        MATNUM, THICK)
         DO 230 LYNUM = 1, NLAYRS
            THICK1 = 0.0
            DETJAC = 0.0
            CALL LYINFO (LYNUM, LTHICK, ZS, MATRL, DCS, NNEL, ELNUM,
     1           NIPXI, NIPETA, NIPSI)
            IF (ITYPE .GT. 300) THEN
               IF (ITYPE.NE.0 .OR. IDENT.NE.0) THEN
                  IF (IFLAG .EQ. 0) THEN
                     THICK1 = 1.0D0
                     CALL ISH3DG (ITYPE, NNEL, IERROR)
                  ELSE IF (IFLAG .EQ. 4) THEN
                     CALL GAUSS (NIPXI, WXI, XI)
                     CALL GAUSS (NIPETA, WETA, ETA)
C                   CALL GAUSS (NIPSI,WSI,SIZ)
                     CALL ISHSHL (ITYPE, NNEL, IERROR)
                     DO 110 K1 = 1, NNEL
                        KP = NOP(K1,ELNUM)
                        ICODE = IAND(ISPB(KP),4)
C
c ---- iF SHELL ROTATIONS ARE ASSEBLED IN THE LOCAL SHELL COORDINATE
c       SYSTEM THEN RETREIVE THE LOCAL z-AXIS FROM STORAGE. ELSE
c       EVALUATE THE NORMAL TO THE SHELL MID PLANE BY A CALL TO THE
c       SHNORM ROUTINE.
C
                        IF (ICODE .GT. 0) THEN
                           CALL SHNORM (ELNUM, K1, NNEL, ITYPE,
     1                        VECT(1,1,K1),VECT(1,2,K1),VECT(1,3
     2                        ,K1))
                        ELSE
                           VECT(1,3,K1) = DBLE(SHELLZ(1,KP))
                           VECT(2,3,K1) = DBLE(SHELLZ(2,KP))
                           VECT(3,3,K1) = DBLE(SHELLZ(3,KP))
                           CALL DIRVEC (VECT(1,1,K1),VECT(1,2,K1)
     1                        ,VECT(1,3,K1))
                        ENDIF
  110                CONTINUE
                  ENDIF
               ENDIF
c           IDENT1 = IDENT
c           ITYPE1 = ITYPE
               DO 150 ISI = 1, NIPSI
                  DO 140 IETA = 1, NIPETA
                     DO 130 IXI = 1, NIPXI
                        INTGPN = (NIPXI*NIPETA)*(ISI-1) + NIPXI*(
     1                     IETA-1) + IXI
                        IF (IFLAG .EQ. 0) THEN
                           CALL JACB3D (INTGPN, ELNUM, NNEL,
     1                        IERROR, DETJAC)
                        ELSE IF (IFLAG .EQ. 4) THEN
                           ISET = 0
                           SIP = SI(INTGPN)
                           CALL GETTHK (INTGPN, ELNUM, NNEL,
```

```
     1                           THICK1, RAD)
                             CALL JACSHL (INTGPN, ELNUM, NNEL,
     1                           THICK1, DETJAC, DUMMY, ISET, SIP)
                         ENDIF
                         CST = DETJAC*W(INTGPN)*THICK1
C*VDIR: IGNORE RECRDEPS
C*VDIR: PREFER VECTOR
                         DO 120 K1 = 1, NNEL
                             M1 = NOP(K1,ELNUM)
                             RX(M1) = RX(M1) + N(K1,INTGPN)*WGTX(
     1                           MATNUM)*CST
                             RY(M1) = RY(M1) + N(K1,INTGPN)*WGTY(
     1                           MATNUM)*CST
                             RZ(M1) = RZ(M1) + N(K1,INTGPN)*WGTZ(
     1                           MATNUM)*CST
  120                    CONTINUE
  130                CONTINUE
  140            CONTINUE
  150        CONTINUE
C
C------ SURFACE LOADS FOR SHELLS - FOLLOWING PORTIONS NOT TESTED.
C
C
             NIPSIT = NIPSI
             NIPSI = 1
             CALL ISHSHL (ITYPE, NNEL, IERROR)
C
             DO 200 ISI = 1, NIPSI
                 DO 190 IETA = 1, NIPETA
                     DO 180 IXI = 1, NIPXI
                         INTGPN = (NIPXI*NIPETA)*(ISI-1) + NIPXI*(
     1                       IETA-1) + IXI
                         ICHK = ISULOD(ELNUM)
                         IF(LYNUM.GT.1.AND.ICHK.EQ.(-1))GO TO170
                         IF (LYNUM.LT.NLAYRS .AND. ICHK.EQ.1)
     1                       GO TO 170
                         IF (ISI.GT.1 .AND. ICHK.EQ.(-1)) GO TO 170
                         IF(ISI.LT.NIPSI.AND.ICHK.EQ.1)GO TO170
                         IF (IFLAG .EQ. 4) THEN
                             ISET = 1
                             IF (ICHK .EQ. 1) SIP = 1.0
                             IF (ICHK .EQ. (-1)) SIP = -1.0
                             IF (ICHK .EQ. 0) GO TO 170
                             CALL GETTHK (INTGPN, ELNUM, NNEL,
     1                           THICK1, RAD)
                             CALL JACSHL (INTGPN, ELNUM, NNEL,
     1                           THICK1, DETJAC, DUMMY, ISET, SIP)
                             DNORM = DSQRT(DUMMY(1)**2+DUMMY(2)**2+
     1                           DUMMY(3)**2)
                             DETJAC = 2.0*DETJAC/THICK1
                             DUMMY(1) = DUMMY(1)/DNORM
                             DUMMY(2) = DUMMY(2)/DNORM
                             DUMMY(3) = DUMMY(3)/DNORM
                             WT = WXI(IXI)*WETA(IETA)
                             DO 160 K1 = 1, NNEL
                             CST1 = DETJAC*WT*PRES(ELNUM,K1)*SIP
                             M1 = NOP(K1,ELNUM)
                             RX(M1) = RX(M1) + N(K1,INTGPN)*DUMMY(1
     1                           )*CST1
                             RY(M1) = RY(M1) + N(K1,INTGPN)*DUMMY(2
     1                           )*CST1
                             RZ(M1) = RZ(M1) + N(K1,INTGPN)*DUMMY(3
     1                           )*CST1
```

```
                              AMX(M1) = AMX(M1) + THICK(K1)*0.5*(
     1                            DUMMY(3)*VECT(2,3,K1)-DUMMY(2)*
     2                            VECT(3,3,K1))
                              AMY(M1) = AMY(M1) + THICK(K1)*0.5*(
     1                            DUMMY(1)*VECT(3,3,K1)-DUMMY(3)*
     2                            VECT(1,3,K1))
                              AMZ(M1) = AMZ(M1) + THICK(K1)*0.5*(
     1                            DUMMY(2)*VECT(1,3,K1)-DUMMY(1)*
     2                            VECT(2,3,K1))
  160                         CONTINUE
                          ENDIF
  170                     CONTINUE
  180                 CONTINUE
  190             CONTINUE
  200         CONTINUE
              NIPSI = NIPSIT
          ELSE
              IF (ITYPE.NE.0 .OR. IDENT.NE.0) CALL ISH2DG (ITYPE,
     1            NNEL, IERROR)
c        IDENT1 = IDENT
c        ITYPE1 = ITYPE
              DO 220 INTGPN = 1, NIP
                  CALL GETTHK (INTGPN, ELNUM, NNEL, THICK1, RAD)
                  CALL JACB2D (INTGPN, ELNUM, NNEL, IERROR, DETJAC)
                  CST = DETJAC*THICK1*W(INTGPN)
C
C*VDIR: IGNORE RECRDEPS
C*VDIR: PREFER VECTOR
C
                  DO 210 K1 = 1, NNEL
                      M1 = NOP(K1,ELNUM)
                      RX(M1)=RX(M1)+N(K1,INTGPN)*WGTX(MATNUM)*CST
                      RY(M1)=RY(M1)+N(K1,INTGPN)*WGTY(MATNUM)*CST
  210             CONTINUE
  220         CONTINUE
          ENDIF
  230     CONTINUE
  240 CONTINUE
C

C --- PLACE RX' S AND RY' S IN THE RIGHT POSITIONS IN THE
C --- LOAD ARRAY.
C
      IF (IDIM .EQ. 2) THEN
C
C*VDIR: PREFER VECTOR
C
          DO 250 K = 1, NNODES
              K1 = NNDF*(K-1) + 1
              K2 = K1 + 1
              R(K1) = R(K1) + RX(K)
              R(K2) = R(K2) + RY(K)
  250     CONTINUE
C
      ELSE IF (IDIM .EQ. 3) THEN
C
C*VDIR: PREFER VECTOR
C
          DO 260 K = 1, NNODES
              K1 = NNDF*(K-1) + 1
              K2 = K1 + 1
              K3 = K2 + 1
              K4 = K3 + 1
```

```
            K5 = K4 + 1
            K6 = K5 + 1
            R(K1) = R(K1) + RX(K)
            R(K2) = R(K2) + RY(K)
            R(K3) = R(K3) + RZ(K)
            R(K4) = R(K4) + AMX(K)
            R(K5) = R(K5) + AMY(K)
            R(K6) = R(K6) + AMZ(K)
  260     CONTINUE
      ENDIF
C
C ----- FOR THE NODES WHICH ARE SHARED BY SHELL ELEMENTS AND THEIR
C       ROTATIONAL DEGREES OF FREEDOM IS ASSEMBLED IN THE LOCAL
C       COORDINATES, TRANSFORM THE LOADS TO THE LOCAL COORDINATES OF
C       THE NODE.
C
      IF (IDIM .EQ. 3) THEN
          DO 300 K1 = 1, NNODES
            I = NNDF*(K1-1)
            ICODE = IAND(ISPB(K1),256)
            ISPS = IAND(ISPB(K1),4)
            I = I + IDIM
            IF (ISPS.EQ.0 .AND. ICODE.GT.0) THEN
                DC(1,3) = DBLE(SHELLZ(1,K1))
                DC(2,3) = DBLE(SHELLZ(2,K1))
                DC(3,3) = DBLE(SHELLZ(3,K1))
                CALL DIRVEC (DC(1,1),DC(1,2),DC(1,3))
C*VDIR: PREFER SCALAR
                DO 280 K2 = 1, IDIM
                    CST = 0.
C*VDIR: PREFER SCALAR
                    DO 270 K3 = 1, IDIM
                        IDIR = I + K3
                        CST = CST + R(IDIR)*DC(K3,K2)
  270               CONTINUE
                    DUMMY(K2) = CST
  280           CONTINUE
C*VDIR: PREFER SCALAR
                DO 290 K2 = 1, IDIM
                    IDIR = I + K2
                    R(IDIR) = DUMMY(K2)
  290           CONTINUE
            ENDIF
  300     CONTINUE
      ENDIF
C
      RETURN
      END
C                     ------------------------------------
C                     *                                  *
C                     *  MULTILAYER NON-LINEAR ANALYSIS  *
C                     *                                  *
C                     ------------------------------------
C
C
C               PROPER(I,K)= 'I'TH PROPERTY OF MATERIAL 'K'.
C               INHISTY= TAPE TO SAVE PREVIOUS HISTORY AT EACH
C                       GAUSS POINT.
C               ITNSPG= TAPE TO SAVE PREVIOUS HISTORY AT EACH
C                       GAUSS POINT.
C               HISTO(I,J,K)= ARRAY TO SAVE 'J' NO. OF DATA FOR
C                             GAUSS POINT 'I' OF THE CURRENT ELE-
C                             MENT IN LAYER 'K'.
```

```
C               CRACK(I,J,K)= ARRAY TO SAVE 'J' NO. OF CRACK DATA FOR
C                             GAUSS POINT 'I' OF THE CURRENT ELE-
C                             MENT IN LAYER 'K'.
C              TENSTF(I,J,K)= ARRAY TO SAVE 'J' NO. OF CRACK DATA FOR
C                             GAUSS POINT 'I' OF THE CURRENT ELE-
C                             MENT IN LAYER 'K'.
C
C
C ------------------------ N  L  P  R  O  P ---------------------------
C
C     INCLUDE(PROCESS)
      SUBROUTINE NLPROP(IOUT,IERROR,MATNUM)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C**************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ELINFO    ELINTM    LYINFO    ISHSHL    GTLTK     GETTHK
C DTRANS    CTNSTF
C
C**************************************************************
      INTEGER ELNUM
      REAL*8 NU,LTHICK
      REAL*4 THICK
      COMMON/INPUT9/THICK(9),IFLAG
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     $              NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC1(10),LDCONT,INCPTR
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON/LPROP/PROPER(25,15)
      COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
      COMMON/TSHEAR/AK1(5000,9),AK2(5000,9),AF1(5000,15,27),
     $              AF2(5000,15,27)
      DIMENSION DSTEEL(6,6),ICOUNT(27,15)
      DO 100 IMATN = 1, MATNUM
C
          IF (PROPER(25,IMATN) .EQ. 1.0) THEN
C
C....            CONCRETE MATERIAL
C
              EI = PROPER(1,IMATN)
              FC = PROPER(5,IMATN)
              EPSC = PROPER(7,IMATN)
              EC = FC/EPSC
              A = EI/EC
              IF (A .LT. 4./3.) THEN
                  WRITE (IOUT, 900) IMATN, A, 'NLPROP CONC. ERROR. 1'
                  STOP
              ENDIF
C
C                CHECK FOR THE LIMITS ON THE POST-CRUSHING PARAM. 'D'.
C
              D = PROPER(9,IMATN)
              IF (A .LE. 2.) THEN
                  C1 = (1.-.5*A)**2.
                  C2 = 1. + A*(A-2.)
                  IF (.NOT.(D.LE.C2 .AND. D.GE.C1)) THEN
                      WRITE (IOUT, 910) IMATN, A, D,
```

```
     1                    'NLPROP. CONC. ERROR. 2'
                      STOP
                  ENDIF
              ENDIF
              IF (A .GT. 2.) THEN
                  IF (.NOT.(D.GE.0. .AND. D.LE.1.)) THEN
                      WRITE (IOUT, 910) IMATN, A, D,
     1                    'NLPROP CONC. ERROR 3'
                      STOP
                  ENDIF
              ENDIF
          ENDIF
C
  100 CONTINUE
C
C
C     GENERATE ELEMENTS OF THE INITIAL CONSTITUTIVE MATRIX
C     D(1,1),D(1,2),D(1,3),D(2,3),D(3,3),D(4,4),D(5,5),D(6,6).RETRIEVE
C     THE DIRECTION COSINES W.R.T GLOBAL XYZ, WHICH
C     INITIALLY IS PUT TO ZERO FOR CONCRETE LAYERS AND IS ALWAY
C     EQUAL TO THE ORIENTATION OF THE STEEL REBARS.THIS IS DONE
C     FOR EACH GAUSS POINT OF EACH ELEMENT AND THE RESULTING DATA
C     FOR EACH BLOCK OF ELEMENTS IS SAVED IN ARRAY 'HISTO'.
C
      IHISTY = 50
      IHIST1 = 60
      IHIST2 = 61
      ITNSPG = 80
      ITNSG1 = 70
      ITNSG2 = 71
C
      REWIND IHISTY
      REWIND IHIST1
      REWIND IHIST2
      REWIND ITNSPG
      REWIND ITNSG1
      REWIND ITNSG2
      DO 240 ELNUM = 1, NELEM
C
C             INITIALIZE 'HISTO' & 'CRACK' FOR EACH BLOCK
C             OF ELEMENTS.
C
          CALL ELINFO (ELNUM, ITYPE, NNEL, IFLAG, ISTART, LINES)
          CALL ELINTM (ELNUM, IDENT, INTCOD, NIPXI, NIPETA, NIPSI,
     1        MATNUM, THICK)
C
          DO 150 LYNUM = 1, 15
              DO 140 IGAUSS = 1, 27
                  ICOUNT(IGAUSS,LYNUM) = 0
                  DO 110 IDATA = 1, 70
                      HISTO(IGAUSS,IDATA,LYNUM) = 0.0
  110             CONTINUE
                  DO 120 IDATA = 1, 250
                      CRACK(IGAUSS,IDATA,LYNUM) = 0.0
  120             CONTINUE
                  DO 130 IDATA = 1, 80
                      TENSTF(IGAUSS,IDATA,LYNUM) = 0.0
  130             CONTINUE
  140         CONTINUE
  150     CONTINUE
C
          DO 230 J = 1, NLAYRS
C
```

```
            CALL LYINFO (J, LTHICK, ZS, MATRL, DCS, NNEL, ELNUM, NIPXI
     1           , NIPETA, NIPSI)
C
            NIP = NIPSI*NIPETA*NIPXI
            IF (IFLAG .EQ. 4) THEN
                IF (NLAYRS .GT. 1) THEN
                    DO KS = 1, 9
                        AK1(ELNUM,KS) = 5.0/6.0
                        AK2(ELNUM,KS) = 5.0/6.0
                    END DO
                    CALL ISHSHL (ITYPE, NNEL, IERROR)
                    DO KS = 1, NIP
                        THICKL = 0.0
                        ZSI = 0.0
                        THICKE = 0.0
                        RAD = 0.0
                        CALL GTLTK (KS, ELNUM, NNEL, THICKL, ZSI,
     1                      LTHICK, ZS)
                        CALL GETTHK (KS, ELNUM, NNEL, THICKE, RAD)
                        AKONST = (3./2.)*(1.0-(4.0*ZSI*ZSI)/(THICKE*
     1                      THICKE))
                        AF1(ELNUM,J,KS) = AKONST
                        AF2(ELNUM,J,KS) = AKONST
                    END DO
                ELSE
                    DO KS = 1, 9
                        AK1(ELNUM,KS) = 5.0/6.0
                        AK2(ELNUM,KS) = 5.0/6.0
                        AF1(ELNUM,1,KS) = 5./6.
                        AF2(ELNUM,1,KS) = 5./6.
                    END DO
                ENDIF
            ENDIF
C
            E = PROPER(1,MATRL)
            NU = PROPER(2,MATRL)
            DO 220 L = 1, NIP
C
C              *******   STEEL LAYER   ********
C
                IF (PROPER(25,MATRL) .EQ. 0.0) THEN
C
C                         ---------------
C                         * STEEL LAYER *
C                         ---------------
C
C
C
C              SAVE D(1,1) AND THEATA=ORIENTATION OF REBARS(RADIANS)
C
C
                    HISTO(L,1,J) = E
                    HISTO(L,14,J) = E
                    HISTO(L,13,J) = PROPER(5,MATRL)
C
C              CALCULATE AND SAVE THE YIELD STRAIN.
C
                    EPSYLD = HISTO(L,13,J)/HISTO(L,14,J)
                    HISTO(L,12,J) = EPSYLD
C
C
C              GENERATE THE CONSTITUTIVE MATRIX FOR THE STEEL
C              IN THE MATERIAL AXES (DIREC. OF REBARS).
```

```
C
C
                    DO 200 IR = 1, 6
                       DO 190 IC = 1, 6
                          DSTEEL(IR,IC) = 0.0
 190                   CONTINUE
 200                CONTINUE
                    DSTEEL(1,1) = E
C
C               ROTATE 'DSTEEL' TO GLOBAL AXES.
C
                    CALL DTRANS (DSTEEL, DCS, IOUT)
C
C    ... STORE STEEL FIBRE DIRECTION
C
                    HISTO(L,17,J) = DCS(1,1)
                    HISTO(L,18,J) = DCS(1,2)
                    HISTO(L,19,J) = DCS(1,3)
C
C               SAVE THE ROTATED CONSTITUTIVE MATRIX IN ARRAY 'CRACK'
C
                    CRACK(L,2,J) = DSTEEL(1,1)
                    CRACK(L,3,J) = DSTEEL(1,2)
                    CRACK(L,4,J) = DSTEEL(1,3)
                    CRACK(L,5,J) = DSTEEL(1,4)
                    CRACK(L,6,J) = DSTEEL(1,5)
                    CRACK(L,7,J) = DSTEEL(1,6)
                    CRACK(L,8,J) = DSTEEL(2,1)
                    CRACK(L,9,J) = DSTEEL(2,2)
                    CRACK(L,10,J) = DSTEEL(2,3)
                    CRACK(L,11,J) = DSTEEL(2,4)
                    CRACK(L,12,J) = DSTEEL(2,5)
                    CRACK(L,13,J) = DSTEEL(2,6)
                    CRACK(L,14,J) = DSTEEL(3,1)
                    CRACK(L,15,J) = DSTEEL(3,2)
                    CRACK(L,16,J) = DSTEEL(3,3)
                    CRACK(L,17,J) = DSTEEL(3,4)
                    CRACK(L,18,J) = DSTEEL(3,5)
                    CRACK(L,19,J) = DSTEEL(3,6)
                    CRACK(L,20,J) = DSTEEL(4,1)
                    CRACK(L,21,J) = DSTEEL(4,2)
                    CRACK(L,22,J) = DSTEEL(4,3)
                    CRACK(L,23,J) = DSTEEL(4,4)
                    CRACK(L,24,J) = DSTEEL(4,5)
                    CRACK(L,25,J) = DSTEEL(4,6)
                    CRACK(L,26,J) = DSTEEL(5,1)
                    CRACK(L,27,J) = DSTEEL(5,2)
                    CRACK(L,28,J) = DSTEEL(5,3)
                    CRACK(L,29,J) = DSTEEL(5,4)
                    CRACK(L,30,J) = DSTEEL(5,5)
                    CRACK(L,31,J) = DSTEEL(5,6)
                    CRACK(L,32,J) = DSTEEL(6,1)
                    CRACK(L,33,J) = DSTEEL(6,2)
                    CRACK(L,34,J) = DSTEEL(6,3)
                    CRACK(L,35,J) = DSTEEL(6,4)
                    CRACK(L,36,J) = DSTEEL(6,5)
                    CRACK(L,37,J) = DSTEEL(6,6)
                    HISTO(L,15,J) = 1.0
                    HISTO(L,16,J) = 1.0
                    GO TO 210
                ENDIF
C
C                      -------------------
```

```
C                      * CONCRETE LAYERS *
C                      -------------------
C
C
                 CONST = E/((1.-2.0*NU)*(1.0+NU))
C
C               CALCULATE D11,D12,D22,D33,D44,D55
C
                 HISTO(L,1,J) = CONST*(1.0-NU)
                 HISTO(L,2,J) = CONST*NU
                 HISTO(L,3,J) = CONST*(1.0-NU)
                 HISTO(L,4,J) = CONST*(1.0-NU)
                 HISTO(L,5,J) = CONST*NU
                 HISTO(L,6,J) = CONST*NU
                 HISTO(L,7,J) = E/(2.0*(1.0+NU))
                 HISTO(L,8,J) = E/(2.0*(1.0+NU))
                 HISTO(L,9,J) = E/(2.0*(1.0+NU))
C
C               HISTO(L,10-18,J)=DIR. COS. FROM GLOBAL XYZ
C
                 HISTO(L,10,J) = 0.0
                 HISTO(L,11,J) = 0.0
                 HISTO(L,12,J) = 0.0
                 HISTO(L,13,J) = 0.0
                 HISTO(L,14,J) = 0.0
                 HISTO(L,15,J) = 0.0
                 HISTO(L,16,J) = 0.0
                 HISTO(L,17,J) = 0.0
                 HISTO(L,18,J) = 0.0
                 HISTO(L,20,J) = E
                 HISTO(L,21,J) = NU
C
C  ... THE COMPRESSION SOFTENING MODEL NEED TO STORE AND UPDATE
C      FC THE COMP. STRENGTH FOR EACH ELEMENT,CONCRETE LAYER AT EACH
C      SAMPLING POINT APPROPRIATELY. SO STORE THIS PARAMETER CORRECTLY.
C
                 HISTO(L,58,J) = PROPER(5,MATRL)
C
C               INITIALLY THE CONSTITUTIVE MATRIX IN THE GLOBAL
C               AXES IS THE SAME AS ONE IN THE MATERIAL AXES.
C
C
C        .....  D11,D12,D13,D14,D15,D16
C
                 CRACK(L,2,J) = HISTO(L,1,J)
                 CRACK(L,3,J) = HISTO(L,2,J)
                 CRACK(L,4,J) = HISTO(L,6,J)
                 CRACK(L,5,J) = 0.0
                 CRACK(L,6,J) = 0.0
                 CRACK(L,7,J) = 0.0
C
C        .....  D21,D22,D23,D24,D25,D26
C
                 CRACK(L,8,J) = HISTO(L,2,J)
                 CRACK(L,9,J) = HISTO(L,3,J)
                 CRACK(L,10,J) = HISTO(L,5,J)
                 CRACK(L,11,J) = 0.0
                 CRACK(L,12,J) = 0.0
                 CRACK(L,13,J) = 0.0
C
C        .....  D31,D32,D33,D34,D35,D36
C
                 CRACK(L,14,J) = HISTO(L,6,J)
```

```
                CRACK(L,15,J) = HISTO(L,5,J)
                CRACK(L,16,J) = HISTO(L,4,J)
                CRACK(L,17,J) = 0.0
                CRACK(L,18,J) = 0.0
                CRACK(L,19,J) = 0.0
C
C       ..... D41,D42,D43,D44,D45,D46
C
                CRACK(L,20,J) = 0.0
                CRACK(L,21,J) = 0.0
                CRACK(L,22,J) = 0.0
                CRACK(L,23,J) = HISTO(L,7,J)
                CRACK(L,24,J) = 0.0
                CRACK(L,25,J) = 0.0
C
C       ..... D51,D52,D53,D54,D55,D56
C
                CRACK(L,26,J) = 0.0
                CRACK(L,27,J) = 0.0
                CRACK(L,28,J) = 0.0
                CRACK(L,29,J) = 0.0
                CRACK(L,30,J) = HISTO(L,8,J)
                CRACK(L,31,J) = 0.0
C
C       ..... D31,D32,D33,34,35,36
C
                CRACK(L,32,J) = 0.0
                CRACK(L,33,J) = 0.0
                CRACK(L,34,J) = 0.0
                CRACK(L,35,J) = 0.0
                CRACK(L,36,J) = 0.0
                CRACK(L,37,J) = HISTO(L,9,J)
                IF (PROPER(25,MATRL) .EQ. 2) GO TO 210
C
C             INITIALLY THERE ARE NO CRACKS AT THE CURRENT POINT.
C             E.G., IF ONE CRACK DEVELOPS THEN CRACK(L,1,J)= 1.
C
                CRACK(L,1,J) = 0.0
                IF (NLAYRS .GT. 1) CALL CTNSTF (ELNUM, L, J, IOUT,
     1              NLAYRS, ICOUNT)
C
C
C
C             HISTO(L,29,J) = 1.  .... STRESS POINT IN PRE-FAILURE
C
C                           = 2.  ....    ;     ;    ; FAILURE STATE
C
C                           = 3.  ....    ;     ;    ; POST-FAILURE
C
C                           =4.   ....    CRUSHED POINT (STIFFNESS=0.)
C
C
C
C             INITIALLY ALL POINTS ARE IN THE PRE-FAILURE STATE.
C
C
                HISTO(L,29,J) = 1.0
C
C
C             HISTO(L,30,J) = 1.   ..... LOADING OF THE STRESS POINT
C
C                           = 2.   ..... UNLOADING   ;    ;      ;
C
```

```
C                              = 3.    .....  RELOADING   ;     ;       ;
C
C
C                  INITIALLY ALL POINTS ARE LOADING (STRAINS INCREASING)
C
C
C SETUP THE TENSION-STIFFENING LAYER INFO IN CONCRETE LAYERS NEXT TO
C STEEL
                 HISTO(L,30,J) = 1.0
  210            CONTINUE
  220        CONTINUE
             WRITE(IHISTY)ELNUM,J,NIP,((CRACK(L,K,J),K=1,250),L=1,NIP)
             WRITE (ITNSPG) ELNUM, J, NIP, MATRL, ((TENSTF(L,K,J),K = 1
     1           ,80), L = 1, NIP), ((HISTO(L,K,J),K = 1,70), L = 1,
     2           NIP), (PROPER(KM,MATRL), KM = 1, 25)
             WRITE(IHIST2)ELNUM,J,NIP,((CRACK(L,K,J),K=1,250),L=1,NIP)
             WRITE (ITNSG2) ELNUM, J, NIP, MATRL, ((TENSTF(L,K,J),K = 1
     1           ,80), L = 1, NIP), ((HISTO(L,K,J),K = 1,70), L = 1,
     2           NIP), (PROPER(KM,MATRL), KM = 1, 25)
  230     CONTINUE
  240 CONTINUE
      RETURN
  900    FORMAT(///,20X,' ERROR',/,30X,'PARAMETER A FOR '
     1          ,'CONCRETE MATRL ',I2,1X,'IS LESS THAN 4./3.',/,
     2          50X,'A=',D10.3)
  910      FORMAT(///,5X,' ERROR',/,30X,'FOR CONCRETE MATRL:',
     1             I2,1X,'A=',D12.5,/,30X,'POST-CRUSHING PARAMETER,D=',
     2             D12.5,1X,'WHICH LIES OUTSIDE ITS ALLOWABLE RANGE')
      END
C=================================================================
C     INCLUDE(PROCESS)
C
      SUBROUTINE CHKRC(INCREM,NIT,IERROR,IOUT,ICHECK)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C**********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ELINFO    ELINTM    LYINFO    ISH3DG    ISHSHL
C ISH2DG    SHNORM    DIRVEC    GETTHK    JACSHL    CHKSTL
C CHKCON    IOGET     GTLTK     DMATCS
C
C**********************************************************
      REAL*4  THICK,SHELLZ
      REAL*8  LTHICK,N,NXI,NETA,NSI,SI
      INTEGER ELNUM,ELNUMB
      CHARACTER*105,DUMMY
      COMMON/INPUTE/ISPB(10000)
      COMMON/SKTR2/SHELLZ(3,10000)
      COMMON/SHLDIR/VECT(3,3,9)
      COMMON/TRANS/V1(3),V2(3),V3(3)
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     $              NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC1(10),LDCONT,INCPTR
      COMMON/ISHAP1/N(20,27),NXI(20,27),NETA(20,27),NSI(20,27),SI(27)
      COMMON/INPUT9/THICK(9),IFLAG
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/LAYERN/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
      COMMON/LPROP/PROPER(25,15)
      COMMON/ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
```

```
     $              ,IHIST2
      COMMON/TSHEAR/AK1(5000,9),AK2(5000,9),AF1(5000,15,27)
     $              ,AF2(5000,15,27)
      COMMON/UTIL1/STRESS(6),STRAIN(6),DUMMY
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/INPUT2/NOP(20,5000)
      COMMON/TSHR1/E1(5000,9),F1(5000,9)
      COMMON/MATER1/DEP(6,6)
      DIMENSION TEMP1(9),TEMP2(9),TEMP3(9),TEMP4(9),AR1(9),AR2(9)
      DIMENSION TEMP1B(9),TEMP2B(9),TEMP3B(9),TEMP4B(9),AR1B(9),AR2B(9)
      DIMENSION TEMP5(15,27),TEMP6(9),TEMP7(15,27),TEMP8(9),TEMPA(9)
      DIMENSION TEMP5B(9),TEMP6B(9),TEMP7B(9),TEMP8B(9)
      DIMENSION TEMPB(9),TEMPC(9),TEMPD(9)
C
C
C        SUBROUTINE CONTROLS THE CONVERGENCE PROCEDURE
C        AND CALLING OF APPROPRIATE SUBROUTINES.
C
C
C FOR EACH ELEMENT, LAYER AND INTEGRATION POINT:
C
C       CONVER = 1.  ....... BOTH 'E' AND 'NU' ARE WITHIN
C                THE TOLERANCE,I.E.,CONVERGENCE IN CURRENT LAYER
C
C       CONVER =999.  ...... EITHER 'E' OR 'NU' FOR AT LEAST
C               ONE GAUSS POINT VIOLATES THE ALLOW. TOLER
C
C       'NSTIFF' DETERMINES IF STIFFNESS MATRIX FOR AT LEAST
C        ONE LAYER OF ELEMENT 'I' IS TO BE UPDATED.
C
C        NSTIFF = 0  ..... NO STIFFNESS UPDATE NECESSARY.
C
C               = 1  .....    STIFFNESS UPDATE NECESSARY.
C
C=====================================================================
C
      ICHECK = 0
      REWIND IHISTY
      REWIND IHIST1
      REWIND IHIST2
      REWIND ITNSPG
      REWIND ITNSG1
      REWIND ITNSG2
      DO 210 ELNUM = 1, NELEM
C
          CALL ELINFO (ELNUM, ITYPE, NNEL, IFLAG, ISTART, LINES)
          CALL ELINTM (ELNUM, IDENT, INTCODE, NIPXI, NIPETA, NIPSI,
     1        MATNUM, THICK)
C
          CONVER = 1.0
          NSTIFF = 0
C
          DO IC = 1, 9
              AR1(IC) = 0.0
              AR2(IC) = 0.0
              TEMP1(IC) = 0.0
              TEMP2(IC) = 0.0
              TEMP3(IC) = 0.0
              TEMP4(IC) = 0.0
              TEMP6(IC) = 0.0
              TEMP8(IC) = 0.0
              TEMPA(IC) = 0.0
              TEMPB(IC) = 0.0
```

```
              TEMPC(IC) = 0.0
              TEMPD(IC) = 0.0
              AR1B(IC) = 0.0
              AR2B(IC) = 0.0
              TEMP1B(IC) = 0.0
              TEMP2B(IC) = 0.0
              TEMP3B(IC) = 0.0
              TEMP4B(IC) = 0.0
              TEMP5B(IC) = 0.0
              TEMP6B(IC) = 0.0
              TEMP7B(IC) = 0.0
              TEMP8B(IC) = 0.0
              TEMPA(IC) = 0.0
              TEMPB(IC) = 0.0
              TEMPC(IC) = 0.0
              TEMPD(IC) = 0.0
          END DO
          DO LYNUM = 1, 15
              DO IC = 1, 27
                  TEMP6(LYNUM,IC) = 0.0
                  TEMP7(LYNUM,IC) = 0.0
              END DO
          END DO
C
          DO 170 LYNUM = 1, NLAYRS
C
              CALL LYINFO (LYNUM, LTHICK, ZS, MATRL, DCS, NNEL, ELNUM,
     1            NIPXI, NIPETA, NIPSI)
C
C
C  RECOVER ARRAY 'HISTO','CRACK','TENSTF' FOR THE CURRENT ELEMENT.
C  THESE ARE REQUIRED FOR THE UPDATING PROCEDURE IN THE CONVERGENCE
C  CRITERION EMPLOYED. STORES ALL THE MATERIAL INFO ABOUT POINT.
C
              NIP = NIPXI*NIPETA*NIPSI
C
              READ (IHISTY) ELNUMB, LYNUMB, NGAUS, ((CRACK(L,K,LYNUM),K
     1             = 1,250), L = 1, NGAUS)
C
              IF (ELNUMB.NE.ELNUM .OR. LYNUMB.NE.LYNUM .OR. NGAUS.NE.NIP
     1            ) THEN
                  WRITE (*, *) 'ERROR READING IHISTY IN CHKRC'
                  WRITE (*, *) 'ELNUM', ELNUM, 'ELNUMB', ELNUMB, 'LYNUM'
     1                , LYNUM, 'LYNUMB', LYNUMB, 'NIP', NIP, 'NGAUS',
     2                NGAUS
                  STOP
              ENDIF
C
              READ (ITNSPG) ELNUMB, LYNUMB, NGAUS, MATRL, ((TENSTF(L,K,
     1            LYNUM),K = 1,80), L = 1, NGAUS), ((HISTO(L,K,LYNUM),K
     2             = 1,70), L = 1, NGAUS), (PROPER(KM,MATRL), KM = 1, 25
     3            )
C
              IF (ELNUMB.NE.ELNUM .OR. LYNUMB.NE.LYNUM .OR. NGAUS.NE.NIP
     1            ) THEN
                  WRITE (*, *) 'ERROR READING ITNSPG IN CHKRC'
                  WRITE (*, *) 'ELNUM', ELNUM, 'ELNUMB', ELNUMB, 'LYNUM'
     1                , LYNUM, 'LYNUMB', LYNUMB, 'NIP', NIP, 'NGAUS',
     2                NGAUS
                  STOP
              ENDIF
C
              IF (ITYPE .LE. 0) GO TO 200
```

```
            IF (ITYPE.NE.0 .OR. IDENT.NE.0) THEN
                IF (ITYPE .GT. 300) THEN
                    IF (IFLAG .EQ. 0) THEN
                        NCB = 3*NNEL
                        CALL ISH3DG (ITYPE, NNEL, IERROR)
                    ELSE IF (IFLAG .EQ. 4) THEN
                        NCB = 6*NNEL
                        CALL ISHSHL (ITYPE. NNEL, IERROR)
                    ENDIF
                ELSE IF (ITYPE .GT. 200) THEN
                    NCB = 2*NNEL
                    CALL ISH2DG (ITYPE, NNEL, IERROR)
                    IF (IFLAG .EQ. 3) THEN
                    ELSE
                    ENDIF
                ENDIF
            ENDIF
C       ITYPE1 = ITYPE
C       IDENT1 = IDENT
C****
            IF (IFLAG .EQ. 4) THEN
                DO 130 K1 = 1, NNEL
                    KP = NOP(K1,ELNUM)
                    ICODE = IAND(ISPB(KP),4)
C
C ---- IF SHELL ROTATIONS ARE ASSEMBLED IN THE LOCAL SHELL COORDINATE
C       SYSTEM THEN RETREIVE THE LOCAL Z-AXIS FROM STORAGE. ELSE
C       EVALUATE THE NORMAL TO THE SHELL MID PLANE BY A CALL TO THE
C       SHNORM ROUTINE.
C
                    IF (ICODE .GT. 0) THEN
                        CALL SHNORM (ELNUM, K1, NNEL, ITYPE, VECT(1,1,
     1                      K1),VECT(1,2,K1),VECT(1,3,K1))
                    ELSE
                        VECT(1,3,K1) = DBLE(SHELLZ(1,KP))
                        VECT(2,3,K1) = DBLE(SHELLZ(2,KP))
                        VECT(3,3,K1) = DBLE(SHELLZ(3,KP))
                        CALL DIRVEC (VECT(1,1,K1),VECT(1,2,K1),VECT(1,
     1                      3,K1))
                    ENDIF
  130           CONTINUE
            ENDIF
C
            DO 160 INTGPN = 1, NIP
                THICKE = 0.0
                IF (IFLAG .EQ. 4) THEN
                    CALL GETTHK (INTGPN, ELNUM, NNEL, THICKE, RAD)
C
C ----- VECTOR V3 RETURNED BY JACSHL IS NORMAL TO MID SURFACE OF THE
C       SHELL AT INTEGRATION POINTS
C
                    ISET = 0
                    SIP = SI(INTGPN)
                    CALL JACSHL (INTGPN, ELNUM, NNEL, THICKE, DETJAC,
     1                  V3, ISET, SIP)
C
C ----- THE COORDINATES VECTORS V1 AND V2 ARE EVALUATED BY DIRVEC
C       WHICH IS PART OF THE ELEMENT LIBRARY MODULE
C
                    CALL DIRVEC (V1, V2, V3)
                ENDIF
C****
C
```

```
C
C         ....  CHECK THE STEEL LAYER
C
C
                IF (PROPER(25,MATRL) .EQ. 0.0) THEN
C
C
C             CHECK THE CONVERGENCE OF THE MATERIAL LAW AT EACH
C             GAUSS POINT OF THE STEEL LAYERS.
C
C
                    TOLER = PROPER(22,MATRL)
                    IF (TOLER .EQ. 0.0) TOLER = 3.0
                    CALL CHKSTL (PROPER(1,MATRL),TOLER,NSTIFF,CONVER,
     1                  ELNUM,INTGPN,ITYPE,IOUT)
C
                ELSE IF (PROPER(25,MATRL) .EQ. 1.0) THEN
C
C
C             CHECK THE CONVERGENCE OF THE MATERIAL LAW AT EACH
C             GAUSS POINT OF THE CONCRETE LAYERS.
C
C
                    TOLER = PROPER(22,MATRL)
                    IF (TOLER .EQ. 0.0) TOLER = 3.0
                    CALL CHKCON (TOLER, ELNUM, IOUT, PROPER(1,MATRL),
     1                  CONVER,NSTIFF,NNEL,INCREM,NIT,ITYPE,INTGPN,
     2                  IERROR,IFLAG)
                ELSE IF (PROPER(25,MATRL) .EQ. 2.0) THEN
                    CALL IOGET (LDEV1, 96, '(A96)', 5)
                ENDIF
C
C ...  COMPUTE THE TRANSVERSE STIFFNESS CORRECTION FACTOR FOR A
C      SHELL ELEMENT ACCOUNTING FOR INHOMOGENEOUS MATERIAL LAYERS
C      AND CRACKING.
C
                IF (IFLAG .EQ. 4) THEN
                    IF (NLAYRS .GT. 1) THEN
                        IPOINT = INTGPN/(NIPXI*NIPETA)
                        INDEX = INTGPN - IPOINT*NIPXI*NIPETA
                        IF (INDEX .EQ. 0) INDEX = NIPXI*NIPETA
C
                        IF (LYNUM.EQ.1 .AND. INTGPN.LE.NIPXI*NIPETA)
     1                      THEN
                            TEMPA(INDEX) = -E1(ELNUM,INDEX)
                            TEMPB(INDEX) = -F1(ELNUM,INDEX)
                        ENDIF
C
C
                        THICKL = 0.0
                        ZSI = 0.0
                        ACONST = 0.0
                        Z1 = 0.0
                        Z2 = 0.0
                        CALL GTLTK (INTGPN, ELNUM, NNEL, THICKL, ZSI,
     1                      LTHICK, ZS)
C
                        IF (NIPSI .EQ. 1) THEN
                            ACONST = THICKL/2.0
                        ELSE IF (NIPSI .EQ. 2) THEN
                            IC = INTGPN/(NIPXI*NIPETA)
                            IP = INTGPN - IC*NIPXI*NIPETA
                            IF(IC.EQ.0.OR.IP.EQ.0.AND.IC.EQ.1)THEN
```

```
                              ACONST = THICKL*0.57735/2.0
                           ELSE IF (IC.EQ.1 .OR. IC.EQ.2 .AND. IP.EQ.
     1                            0) THEN
                              ACONST = THICKL*0.42265/2.0
                           ENDIF
                        ELSE IF (NIPSI .EQ. 3) THEN
                           IC = INTGPN/(NIPXI*NIPETA)
                           IP = INTGPN - IC*NIPXI*NIPETA
                           IF(IC.EQ.0.OR.IC.EQ.1.AND.IP.EQ.0)THEN
                              ACONST = THICKL*0.7745966/2.0
                           ELSE IF (IC.EQ.1 .OR. IP.EQ.0 .AND. IC.EQ.
     1                            2) THEN
                              ACONST = THICKL/2.0
                           ELSE IF (IC.EQ.2 .OR. IP.EQ.0 .AND. IC.EQ.
     1                            3) THEN
                              ACONST = THICKL*0.2254034/2.0
                           ENDIF


                        ENDIF
C
                        DO IRS = 1, 6
                           DO IPS = 1, 6
                              DEP(IRS,IPS) = 0.0
                           END DO
                        END DO
C
                        ICODE = 0
                        IPG = 1
                        CALL DMATCS (ELNUM, ITYPE, INTGPN, IFLAG, IOUT
     1                      , ICODE, IPG)
C
                        DEP(5,5) = DEP(5,5)/(AF2(ELNUM,LYNUM,INTGPN)*
     1                      5./6.)
                        DEP(6,6) = DEP(6,6)/(AF1(ELNUM,LYNUM,INTGPN)*
     1                      5./6.)
C
                        Z1 = (-E1(ELNUM,INDEX)) + ZSI + THICKE/2.0
                        Z2 = (-F1(ELNUM,INDEX)) + ZSI + THICKE/2.0
C
C
C TRANSVERSE COMPONENTS IN THE X'DIRECTION
C
C     COMPUTE THE R1 TERM
C
                        AR1(INDEX) = DEP(1,1)*(1/3.)*(Z1**3-TEMPA(
     1                     INDEX)**3) + AR1(INDEX) + AR1B(INDEX)
                        AR1B(INDEX) = DEP(1,1)*(1/3.)*((Z1+ACONST)**3-
     1                     Z1**3)
C
C     COMPUTE THE G TERM, GBAR TERM AND ACCUMALATE
C
                        TEMP5(LYNUM,INTGPN) = TEMPC(INDEX) + ((-DEP(1,
     1                     1)*(1/2.)*(Z1**2-TEMPA(INDEX)**2)))
                        TEMP5B(INDEX) = -DEP(1,1)*(1/2.)*((Z1+ACONST)
     1                     **2-Z1**2)
                        TEMP6(INDEX) = TEMPC(INDEX)*(Z1-TEMPA(INDEX))/
     1                     THICKE - DEP(1,1)*((Z1**3-TEMPA(INDEX)**3)
     2                     /6.-(Z1-TEMPA(INDEX))*TEMPA(INDEX)**2/2.)/
     3                     THICKE + TEMP6B(INDEX) + TEMP6(INDEX)
                        TEMP6B(INDEX) = TEMP5(LYNUM,INTGPN)*ACONST/
     1                     THICKE - DEP(1,1)*(((Z1+ACONST)**3-Z1**3)/
     2                     6.-ACONST*Z1**2/2.)/THICKE
C
```

```
C         COMPUTE THE HG TERM
C
                       IF (PROPER(25,MATRL) .NE. 0.0) THEN
                          TEMP1(INDEX) = TEMP1(INDEX) + DEP(6,6)*(Z1
     1                        -TEMPA(INDEX)) + TEMP1B(INDEX)
                          TEMP1B(INDEX) = DEP(6,6)*ACONST
                          XX1 = Z1 - TEMPA(INDEX)
                          XX2 = Z1**2 - TEMPA(INDEX)**2
                          XX3 = Z1**3 - TEMPA(INDEX)**3
                          XX4 = Z1**4 - TEMPA(INDEX)**4
                          XX5 = Z1**5 - TEMPA(INDEX)**5
                          T1 = TEMPC(INDEX)*TEMPC(INDEX)*XX1
                          T2 = (TEMPA(INDEX)**4*DEP(1,1)*XX1)/4.0
                          T3 = (XX5*DEP(1,1))/20.0
                          T4 = (TEMPA(INDEX)**2*XX3*DEP(1,1))/6.0
                          T5 = TEMPC(INDEX)*TEMPA(INDEX)**2*XX1
                          T6 = TEMPC(INDEX)*XX3/3.0
                          TEMP3(INDEX) = TEMP3(INDEX) + (T1+DEP(1,1)
     1                        *(T5-T6+T2+T3-T4))/DEP(6,6) + TEMP3B(
     2                        INDEX)
                          XX1B = ACONST
                          XX2B = (Z1+ACONST)**2 - Z1**2
                          XX3B = (Z1+ACONST)**3 - Z1**3
                          XX4B = (Z1+ACONST)**4 - Z1**4
                          XX5B = (Z1+ACONST)**5 - Z1**5
                          T1B = TEMP5(LYNUM,INTGPN)*TEMP5(LYNUM,
     1                        INTGPN)*XX1
                          T2B = (Z1**4*DEP(1,1)*XX1B)/4.0
                          T3B = (XX5B*DEP(1,1))/20.0
                          T4B = (Z1**2*XX3B*DEP(1,1))/6.0
                          T5B = TEMP5(LYNUM,INTGPN)*Z1**2*XX1B
                          T6B = TEMP5(LYNUM,INTGPN)*XX3B/3.0
                          TEMP3B(INDEX) = (T1B+DEP(1,1)*(T5B-T6B+T2B
     1                        +T3B-T4B))/DEP(6,6)
C     ELSE
C
C         COMPUTE THE HG TERM
C
C       IF(DEP(1,1).NE.0.0) THEN
C         TEMP1(INDEX) = TEMP1(INDEX)+(DEP(1,1)*0.5*
C    $                 (Z1-TEMPA(INDEX)))
C         XX1= Z1-TEMPA(INDEX)
C         XX2= (Z1**2)-(TEMPA(INDEX)**2)
C         XX3= (Z1**3)-(TEMPA(INDEX)**3)
C         XX4= (Z1**4)-(TEMPA(INDEX)**4)
C         XX5= (Z1**5)-(TEMPA(INDEX)**5)
C         T1 = TEMPC(INDEX)*TEMPC(INDEX)*XX1
C         T2 = ((TEMPA(INDEX)**4)*DEP(1,1)*XX1)/4.0
C         T3 = (XX5*DEP(1,1))/20.0
C         T4 = ((TEMPA(INDEX)**2)*XX3*DEP(1,1))/6.0
C         T5 = TEMPC(INDEX)*(TEMPA(INDEX)**2)*XX1
C         T6 = TEMPC(INDEX)*XX3/3.0
C         TEMP3(INDEX) = TEMP3(INDEX) +((T1+(DEP(1,1)*(T5-T6+T2+T3-T4))
C    $                 )/(DEP(1,1)*0.5))
C       END IF
                       ENDIF
                       TEMPC(INDEX) = TEMP5(LYNUM,INTGPN) + TEMP5B(
     1                    INDEX)
C
C
C TRANSVERSE COMPONENTS IN THE Y'DIRECTION
C
C
```

```
C     COMPUTE THE R2 TERM
C
                          AR2(INDEX) = DEP(2,2)*(1/3.)*(Z2**3-TEMPB(
     1                       INDEX)**3) + AR2(INDEX) + AR2B(INDEX)
                          AR2B(INDEX) = DEP(2,2)*(1/3.)*((Z2+ACONST)**3-
     1                       Z2**3)
C
C     COMPUTE THE G TERM, GBAR TERM AND ACCUMALATE
C
                          TEMP7(LYNUM,INTGPN) = TEMPD(INDEX) + ((-DEP(2,
     1                       2)*(1/2.)*(Z2**2-TEMPB(INDEX)**2)))
                          TEMP7B(INDEX) = -DEP(2,2)*(1/2.)*((Z2+ACONST)
     1                       **2-Z2**2)
                          TEMP8(INDEX) = TEMPD(INDEX)*(Z2-TEMPB(INDEX))/
     1                       THICKE - DEP(2,2)*((Z2**3-TEMPB(INDEX)**3)
     2                       /6.-(Z2-TEMPB(INDEX))*TEMPB(INDEX)**2/2.)/
     3                       THICKE + TEMP8(INDEX) + TEMP8B(INDEX)
                          TEMP8B(INDEX) = TEMP7(LYNUM,INTGPN)*ACONST/
     1                       THICKE - DEP(2,2)*(((Z2+ACONST)**3-Z2**3)/
     2                       6.-ACONST*Z2**2/2.)/THICKE
C
C         COMPUTE THE HG TERM
C
                          IF (PROPER(25,MATRL) .NE. 0.0) THEN
                             TEMP2(INDEX) = TEMP2(INDEX) + DEP(5,5)*(Z2
     1                          -TEMPB(INDEX)) + TEMP2B(INDEX)
                             TEMP2B(INDEX) = DEP(5,5)*ACONST
                             YY1 = Z2 - TEMPB(INDEX)
                             YY2 = Z2**2 - TEMPB(INDEX)**2
                             YY3 = Z2**3 - TEMPB(INDEX)**3
                             YY4 = Z2**4 - TEMPB(INDEX)**4
                             YY5 = Z2**5 - TEMPB(INDEX)**5
                             T1 = TEMPD(INDEX)*TEMPD(INDEX)*YY1
                             T2 = (TEMPB(INDEX)**4*DEP(2,2)*YY1)/4.0
                             T3 = (YY5*DEP(2,2))/20.0
                             T4 = (TEMPB(INDEX)**2*YY3*DEP(2,2))/6.0
                             T5 = TEMPD(INDEX)*TEMPB(INDEX)**2*YY1
                             T6 = TEMPD(INDEX)*YY3/3.0
                             TEMP4(INDEX) = TEMP4(INDEX) + (T1+DEP(2,2)
     1                          *(T5-T6+T2+T3-T4))/DEP(5,5) + TEMP4B(
     2                          INDEX)
                             YY1B = ACONST
                             YY2B = (Z2+ACONST)**2 - Z2**2
                             YY3B = (Z2+ACONST)**3 - Z2**3
                             YY4B = (Z2+ACONST)**4 - Z2**4
                             YY5B = (Z2+ACONST)**5 - Z2**5
                             T1B = TEMP7(LYNUM,INTGPN)*TEMP7(LYNUM,
     1                          INTGPN)*YY1B
                             T2B = (Z2**4*DEP(2,2)*YY1B)/4.0
                             T3B = (YY5B*DEP(2,2))/20.0
                             T4B = (Z2**2*YY3B*DEP(2,2))/6.0
                             T5B = TEMP7(LYNUM,INTGPN)*Z2**2*YY1B
                             T6B = TEMP7(LYNUM,INTGPN)*YY3B/3.0
                             TEMP4B(INDEX) = (T1B+DEP(2,2)*(T5B-T6B+T2B
     1                          +T3B-T4B))/DEP(5,5)
C     ELSE
C
C         COMPUTE THE HG TERM
C
C       IF(DEP(2,2).NE.0.0) THEN
C         TEMP2(INDEX) = TEMP2(INDEX)+(DEP(2,2)*0.5*
C    $                   (Z2-TEMPB(INDEX)))
C         YY1= Z2-TEMPB(INDEX)
```

```
C          YY2= (Z2**2)-(TEMPB(INDEX)**2)
C          YY3= (Z2**3)-(TEMPB(INDEX)**3)
C          YY4= (Z2**4)-(TEMPB(INDEX)**4)
C          YY5= (Z2**5)-(TEMPB(INDEX)**5)
C          T1 = TEMPD(INDEX)*TEMPD(INDEX)*YY1
C          T2 = ((TEMPB(INDEX)**4)*DEP(2,2)*YY1)/4.0
C          T3 = (YY5*DEP(2,2))/20.0
C          T4 = ((TEMPB(INDEX)**2)*YY3*DEP(2,2))/6.0
C          T5 = TEMPD(INDEX)*(TEMPB(INDEX)**2)*YY1
C          T6 = TEMPD(INDEX)*YY3/3.0
C          TEMP4(INDEX) = TEMP4(INDEX) +((T1+(DEP(2,2)*(T5-T6+T2+T3-T4))
C     $                   )/(DEP(2,2)*0.5))
C        END IF
                          ENDIF
                          TEMPD(INDEX) = TEMP7(LYNUM,INTGPN) + TEMP7B(
     1                        INDEX)
C
C
                          TEMPA(INDEX) = Z1 + ACONST
                          TEMPB(INDEX) = Z2 + ACONST
C
                      ENDIF
                  ENDIF
  160         CONTINUE
C
C      THE UPDATED 'HISTO' , 'CRACK' & 'TENSTF' ARRAYS OF THE CURRENT
C      BLOCK OF ELEMENTS IS STORED
C
C
              WRITE (IHIST1) ELNUM, LYNUM, NIP, ((CRACK(L,K,LYNUM),K = 1
     1            ,250), L = 1, NIP)
              WRITE (ITNSG1) ELNUM, LYNUM, NIP, MATRL, ((TENSTF(L,K,
     1            LYNUM),K = 1,80), L = 1, NIP), ((HISTO(L,K,LYNUM),K =
     2            1,70), L = 1, NIP), (PROPER(KM,MATRL), KM = 1, 25)
C
  170     CONTINUE
C
C .. FOR EACH ELEMENT GPNT VERTICAL LINE COMPUTE THE K1 AND K2 FACTOR
C .. FOR EACH GPNT IN A VERTICAL LINE COMPUTE THE F1 AND F2 FACTOR
C
          IF (IFLAG .EQ. 4) THEN
              IF (NLAYRS .GT. 1) THEN
                  DO LY = 1, NLAYRS
                      CALL LYINFO (LY, LTHICK, ZS, MATRL, DCS, NNEL,
     1                    ELNUM, NIPXI, NIPETA, NIPSI)
                      NIP1 = NIPXI*NIPETA*NIPSI
                      II = NIPXI*NIPETA
                      IF (PROPER(25,MATRL) .NE. 0.0) THEN
                          DO IPN = 1, NIP1
                              IPOINT = IPN/II
                              INDEX = IPN - IPOINT*II
                              IF (INDEX .EQ. 0) INDEX = II
                              AK1(ELNUM,INDEX) = AR1(INDEX)**2/(TEMP1(
     1                            INDEX)*TEMP3(INDEX))
                              AK2(ELNUM,INDEX) = AR2(INDEX)**2/(TEMP2(
     1                            INDEX)*TEMP4(INDEX))
                              CALL GETTHK(IPN,ELNUM,NNEL,THICKE,RAD)
                              AF1(ELNUM,LY,IPN) = TEMP5(LY,IPN)/TEMP6(
     1                            INDEX)
                              AF2(ELNUM,LY,IPN) = TEMP7(LY,IPN)/TEMP8(
     1                            INDEX)
                          END DO
                      ENDIF
```

```
               END DO
            ENDIF
         ENDIF
         IF (CONVER .EQ. 999.0) ICHECK = 1
  200    CONTINUE
  210 CONTINUE
C
      ITEMP = IHISTY
      IHISTY = IHIST1
      IHIST1 = ITEMP
      ITEMP = ITNSPG
      ITNSPG = ITNSG1
      ITNSG1 = ITEMP
C
      IF (ICHECK .EQ. 0) THEN
         REWIND IHISTY
         REWIND IHIST1
         REWIND IHIST2
         REWIND ITNSPG
         REWIND ITNSG1
         REWIND ITNSG2
C
         ICP = 62
         REWIND ICP
C
         DO 230 ELNUM = 1, NELEM
C
            CALL ELINFO (ELNUM, ITYPE, NNEL, IFLAG, ISTART, LINES)
            CALL ELINTM (ELNUM, IDENT, INTCODE, NIPXI, NIPETA, NIPSI,
     1           MATNUM, THICK)
C
            DO 220 LYNUM = 1, NLAYRS
C
               CALL LYINFO (LYNUM, LTHICK, ZS, MATRL, DCS, NNEL,
     1              ELNUM, NIPXI, NIPETA, NIPSI)
C
C
C  RECOVER ARRAY 'HISTO','CRACK','TENSTF' FOR THE CURRENT ELEMENT.
C  THESE ARE REQUIRED FOR THE UPDATING PROCEDURE IN THE CONVERGENCE
C  CRITERION EMPLOYED. STORES ALL THE MATERIAL INFO ABOUT POINT.
C
               NIP = NIPXI*NIPETA*NIPSI
C
               READ (IHISTY) ELNUMB, LYNUMB, NGAUS, ((CRACK(L,K,LYNUM
     1              ),K = 1,250), L = 1, NGAUS)
C
               IF (ELNUMB.NE.ELNUM .OR. LYNUMB.NE.LYNUM .OR. NGAUS
     1              .NE.NIP) THEN
                  WRITE (*, *) 'ERROR READING IHISTY IN CHKRC'
                  WRITE (*, *) 'ELNUM', ELNUM, 'ELNUMB', ELNUMB,
     1                 'LYNUM', LYNUM, 'LYNUMB', LYNUMB, 'NIP', NIP,
     2                 'NGAUS', NGAUS
                  STOP
               ENDIF
C
               READ (ITNSPG) ELNUMB, LYNUMB, NGAUS, MATRL, ((TENSTF(L
     1              ,K,LYNUM),K = 1,80), L = 1, NGAUS), ((HISTO(L,K,
     2              LYNUM),K = 1,70), L = 1, NGAUS), (PROPER(KM,MATRL)
     3              , KM = 1, 25)
C
               IF (ELNUMB.NE.ELNUM .OR. LYNUMB.NE.LYNUM .OR. NGAUS
     1              .NE.NIP) THEN
                  WRITE (*, *) 'ERROR READING ITNSPG IN CHKRC'
```

```
                 WRITE (*, *) 'ELNUM', ELNUM, 'ELNUMB', ELNUMB,
     1                'LYNUM', LYNUM, 'LYNUMB', LYNUMB, 'NIP', NIP,
     2                'NGAUS', NGAUS
                 STOP
              ENDIF
C
C     THE UPDATED 'HISTO' , 'CRACK' & 'TENSTF' ARRAYS OF THE CURRENT
C     BLOCK OF ELEMENTS IS STORED
C
C
              WRITE (IHIST2) ELNUM, LYNUM, NIP, ((CRACK(L,K,LYNUM),K
     1             = 1,250), L = 1, NIP)
              WRITE (ITNSG2) ELNUM, LYNUM, NIP, MATRL, ((TENSTF(L,K,
     1           LYNUM),K = 1,80), L = 1, NIP), ((HISTO(L,K,LYNUM),
     2           K = 1,70), L = 1, NIP), (PROPER(KM,MATRL), KM = 1
     3           , 25)
C
              WRITE (ICP) ELNUM, LYNUM, NIP, (AF1(ELNUM,LYNUM,KJ),
     1           KJ = 1, NIP), (AF2(ELNUM,LYNUM,KJ), KJ = 1, NIP)
  220        CONTINUE
  230     CONTINUE
      ENDIF
      RETURN
      END
C
C
C ------------------------- C H K C O N ----------------------
C
C     INCLUDE(PROCESS)
      SUBROUTINE CHKCON(TOLER,IELEM,IOUT,PROPER,CONVER,NSTIFF,NNEL
     $,INCREM,NIT,ITYPE,L,IERROR,IFLAG)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C IOGET     PSTSTN    OTOSEN    CRKPNT    TSSTIF    CMPSFT
C SUBCON    CON23     CON12     STATUS    UNRELD    UPDMAT
C ROCRAK
C
C************************************************************
      REAL*8 LTHICK
      CHARACTER*105,DUMMY
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON /BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON /ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
      COMMON/UTIL1/STRESS(6),STRAIN(6),DUMMY
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/TRANS/V1(3),V2(3),V3(3)
      COMMON/MATER1/DEP(6,6)
      COMMON/ICKPIF/ICKFGO,ICKNTR
      DIMENSION PROPER(25),EPSILN(6),PVAL(3),PDIR(3,3),PSTDIR(3,3)
     $        ,DTSTIF(6,6),TEMP11(6,6)
       LOGICAL CHKGUS
C
C
C      SUBROUTINE TO CHECK CONVERGENCE FOR EACH GAUSS POINT OF
C      CONCRETE LAYER 'J' OF ELEMENT 'IELEM'.IF NOT CONVERGED FOR
C      ANY GAUSS POINT THE MATERIAL PARAMETERS WILL BE MODIFIED
```

```
C      APPROPRIATELY AND THE NEW ELEMENT STIFFNESS WILL BE
C      CALCULATED.
C
C
C
C
C              'CHKGUS' DETERMINES IF THE MATERIAL LAW AT THE CURRENT
C              GAUSS POINT HAS CONVERGED.
C
C              CHKGUS = .FALSE.     .....  CONVERGENCE
C
C                     = .TRUE.      .....  SHOULD UPDATE THE 'D' MATRIX
      MAXCRK = 10
      MAXCK2 = 30
C
      CHKGUS = .FALSE.
      IFN = PROPER(24)
      IFC = PROPER(23)
      EPSMAX = 0.0
      EPSMIN = 0.0
      EPSMID = 0.0
      EPS1 = 0.0
      EPS2 = 0.0
      EPS3 = 0.0
      EPSX = 0.0
      EPSY = 0.0
      EPSZ = 0.0
      GAMAXY = 0.0
      GAMAYZ = 0.0
      GAMAXZ = 0.0
      SIGMA1 = 0.0
      SIGMA2 = 0.0
      SIGMA3 = 0.0
      ICOUP = 0
      IFLG1 = 0
      IFLG2 = 0
      IFLG3 = 0
      ITCRK = 0
      D11 = 0.0
      D12 = 0.0


      D13 = 0.0
      D21 = 0.0
      D22 = 0.0
      D23 = 0.0
      D31 = 0.0
      D32 = 0.0
      D33 = 0.0
      PERCNT = 0.0
      J = LYNUM
C
      CALL IOGET (LDEV1, 96, '(A96)', 5)
C
      EPSX = STRAIN(1)
      EPSY = STRAIN(2)
      EPSZ = STRAIN(3)
      GAMAXY = STRAIN(4)
      GAMAYZ = STRAIN(5)
      GAMAXZ = STRAIN(6)
      GHI = DMAX1(DMAX1(DMAX1(DMAX1(DMAX1(DABS(EPSX),DABS(EPSY)),DABS(
     1    EPSZ)),DABS(GAMAXY)),DABS(GAMAYZ)),DABS(GAMAXZ))
      IF (DABS(EPSX) .LT. 1.E-3*GHI) EPSX = 0.0
      IF (DABS(EPSY) .LT. 1.E-3*GHI) EPSY = 0.0
```

```
      IF (DABS(EPSZ) .LT. 1.E-3*GHI) EPSZ = 0.0
      IF (DABS(GAMAXY) .LT. 1.E-3*GHI) GAMAXY = 0.0
      IF (DABS(GAMAYZ) .LT. 1.E-3*GHI) GAMAYZ = 0.0
      IF (DABS(GAMAXZ) .LT. 1.E-3*GHI) GAMAXZ = 0.0
C
C
C                MAXCRK = MAX. ALLOWABLE CRACKS AT THE POINT
C
C
C
C       IF THERE EXIST CRACKS AT THIS POINT CHECK THE
C       CRACKED POINT'S STATUS ; CALCULATE PRINCIPAL
C       STRAINS,'EPS1','EPS2' AND EPS3 FOR INTACT CONCRETE
C       AND DETERMINE PRINCIPAL STRESSES 'SIGMA1' ,'SIGMA2' &
C       'SIGMA3'.
C
      IF (CRACK(L,1,J) .GE. 1.) THEN
C
C
C          WITH THE GLOBAL STRAINS AT THE GAUSS POINT OF THE
C          CURRENT LAYER CALCULATE AND SAVE THE MAX. PRINCI.
C          STRAIN DIRECTION,'PRIDIR', FOR THE CRACKED POINT.
C
C
          EPSILN(1) = EPSX
          EPSILN(2) = EPSY
          EPSILN(3) = EPSZ
          EPSILN(4) = GAMAXY
          EPSILN(5) = GAMAYZ
          EPSILN(6) = GAMAXZ
C
          IFG = 1
          CALL PSTSTN (EPSILN, PVAL, PSTDIR, IFG, IOUT)
C
          EPSMAX = PVAL(1)
          EPSMID = PVAL(2)
          EPSMIN = PVAL(3)
          AEP = DMAX1(DMAX1(DABS(EPSMAX),DABS(EPSMID)),DABS(EPSMIN))
          IF (DABS(EPSMAX) .LT. 1E-3*AEP) EPSMAX = 0.0
          IF (DABS(EPSMID) .LT. 1E-3*AEP) EPSMID = 0.0
          IF (DABS(EPSMIN) .LT. 1E-3*AEP) EPSMIN = 0.0
C
C STRENGTH REDUCTION DUE TO CRACKING,AND RECOMPUTATION OF PARAMETERS
C FOR THE OTTOSEN ENVELOPE. THE TOTAL STRAINS RECOVERED FOR UPDATING
C THE MATERIAL MODEL IF CONVERGENCE IS NOT OBSERVED POINTWISE.
C
          FC = PROPER(5)
          FT = PROPER(6)
          IF (CRACK(L,1,J) .GE. 1.0) THEN
              IF (IFC .EQ. 2) THEN
                  IF (EPSMAX .LT. PROPER(7)) THEN
                      FC = FC*(1.0-0.2*(EPSMAX/PROPER(7)))
                  ELSE
                      FC = 0.8*FC
                  ENDIF
              ENDIF
          ENDIF
C
          FC = DMIN1(HISTO(L,58,J),FC)
          HISTO(L,58,J) = FC
C
          CALL OTOSEN (FC, FT, PROPER, IOUT)
C
```

```
C
         CALL CRKPNT (EPSX, EPSY, EPSZ, GAMAXY, GAMAYZ, GAMAXZ, PROPER
     1       , TOLER, CHKGUS, NSTIFF, CONVER, IOUT, EPS1, EPS2, EPS3,
     2       IELEM, MAXCRK, MAXCK2, SIGMA1, SIGMA2, SIGMA3, L, J, PDIR
     3       , NLAYRS, NNEL, ITYPE, IFLAG, ISOFT, ITCRK)
C
         AEP = DMAX1(DMAX1(DABS(EPS1),DABS(EPS2)),DABS(EPS3))
         IF (DABS(EPS1) .LT. 1E-3*AEP) EPS1 = 0.0
         IF (DABS(EPS2) .LT. 1E-3*AEP) EPS2 = 0.0
         IF (DABS(EPS3) .LT. 1E-3*AEP) EPS3 = 0.0
C
C      PRINCIPAL STRESSS &  DIRECTION OF INTACT CONCRETE ARE STORED.
C      THIS WILL ALSO BE THE MAX. PRINCIPAL STRESS
C      DIRECTION FOR THE CRACKED POINT.
C
         SIGMA1 = DMIN1(PROPER(6),SIGMA1)
         SIGMA2 = DMIN1(PROPER(6),SIGMA2)
         SIGMA3 = DMIN1(PROPER(6),SIGMA3)
         IF (IFLAG .EQ. 4) THEN
             DOT1 = V3(1)*PDIR(1,1) + V3(2)*PDIR(1,2) + V3(3)*PDIR(1,3)
             DOT2 = V3(1)*PDIR(2,1) + V3(2)*PDIR(2,2) + V3(3)*PDIR(2,3)
             DOT3 = V3(1)*PDIR(3,1) + V3(2)*PDIR(3,2) + V3(3)*PDIR(3,3)
             IF (DABS(DOT1) .GE. 0.99) SIGMA1 = 0.0
             IF (DABS(DOT2) .GE. 0.99) SIGMA2 = 0.0
             IF (DABS(DOT3) .GE. 0.99) SIGMA3 = 0.0
         ENDIF
         HISTO(L,61,J) = SIGMA1
         HISTO(L,62,J) = SIGMA2
         HISTO(L,63,J) = SIGMA3
C
         HISTO(L,10,J) = PDIR(1,1)
         HISTO(L,11,J) = PDIR(1,2)
         HISTO(L,12,J) = PDIR(1,3)
         HISTO(L,13,J) = PDIR(2,1)
         HISTO(L,14,J) = PDIR(2,2)
         HISTO(L,15,J) = PDIR(2,3)
         HISTO(L,16,J) = PDIR(3,1)
         HISTO(L,17,J) = PDIR(3,2)
         HISTO(L,18,J) = PDIR(3,3)
C
C THE NEW TENSION STIFFENING MODEL PARAMETERS AND THE COUPLING DATA
C  ARE RECOMPUTED IF NECESSARY.
C
         CALL TSSTIF (EPSX, EPSY, EPSZ, GAMAXY, GAMAYZ, GAMAXZ, SIGMA1
     1       , SIGMA2, SIGMA3, DTSTIF, L, J, CONVER, NSTIFF, CHKGUS,
     2       PROPER, TOLER, IELEM, NNEL, IFN, IFC, ICOUP, IFLG1, IFLG2
     3       , IFLG3, ISOFT, ITCRK, TEMP11)
C
         HISTO(L,44,J) = PSTDIR(1,1)
         HISTO(L,45,J) = PSTDIR(1,2)
         HISTO(L,46,J) = PSTDIR(1,3)
         HISTO(L,47,J) = PSTDIR(2,1)
         HISTO(L,48,J) = PSTDIR(2,2)
         HISTO(L,49,J) = PSTDIR(2,3)
         HISTO(L,50,J) = PSTDIR(3,1)
         HISTO(L,51,J) = PSTDIR(3,2)
         HISTO(L,52,J) = PSTDIR(3,3)
         GO TO 120
      ENDIF
C
      FC = PROPER(5)
      FT = PROPER(6)
C     WRITE(*,*) 'FT',FT
```

```
C
      CALL OTOSEN (FC, FT, PROPER, IOUT)
C
C        DETERMINE PRINCIPAL STRAINS AND DIRECTION COSINES OF
C        STRAIN TENSOR, BEFORE CRACKING OCCOURS AT THE POINT.
C
C
      EPSILN(1) = EPSX
      EPSILN(2) = EPSY
      EPSILN(3) = EPSZ
      EPSILN(4) = GAMAXY
      EPSILN(5) = GAMAYZ
      EPSILN(6) = GAMAXZ
C
      IFG = 1
      CALL PSTSTN (EPSILN, PVAL, PDIR, IFG, IOUT)
      EPS1 = PVAL(1)
      EPS2 = PVAL(2)
      EPS3 = PVAL(3)
C
      AEP = DMAX1(DMAX1(DABS(EPS1),DABS(EPS2)),DABS(EPS3))
      IF (DABS(EPS1) .LT. 1E-3*AEP) EPS1 = 0.0
      IF (DABS(EPS2) .LT. 1E-3*AEP) EPS2 = 0.0
      IF (DABS(EPS3) .LT. 1E-3*AEP) EPS3 = 0.0
C
      IF (HISTO(L,29,J) .EQ. 4.) THEN
C
C               IF THE POINT IS CRUSHED SAVE THE CURRENT STRAINS ONLY.
C
          HISTO(L,22,J) = EPS1
          HISTO(L,23,J) = EPS2
          HISTO(L,24,J) = EPS3
C
          HISTO(L,10,J) = PDIR(1,1)
          HISTO(L,11,J) = PDIR(1,2)
          HISTO(L,12,J) = PDIR(1,3)
          HISTO(L,13,J) = PDIR(2,1)
          HISTO(L,14,J) = PDIR(2,2)
          HISTO(L,15,J) = PDIR(2,3)
          HISTO(L,16,J) = PDIR(3,1)
          HISTO(L,17,J) = PDIR(3,2)
          HISTO(L,18,J) = PDIR(3,3)
C
C               CHECK THE NEXT GAUSS POINT
C
          RETURN
      ENDIF
C
C             USING THE EXISTING CONSTITUTIVE MATRIX DETERMINE
C             PRINCIPAL STRESSES; ASSUME THAT PRINCIPAL STRESS
C             AND STRAIN DIRECTIONS COINCIDE(FOR UNCRACKED
C             PORTION OF THE CONCRETE).
C
C
      DO IM = 1, 6
          DO IN = 1, 6
              DEP(IM,IN) = 0.0
          END DO
      END DO
C
C RECOVER THE SOLID/SHELL STIFFNESS MATRIX FOR THE ELEMENT AND COMPUTE
C THE PRINCIPAL STRESSES.
C
```

```
      D11 = HISTO(L,1,J)
      D12 = HISTO(L,2,J)
      D13 = HISTO(L,5,J)
      D21 = HISTO(L,2,J)
      D22 = HISTO(L,3,J)
      D23 = HISTO(L,6,J)
      D31 = HISTO(L,5,J)
      D32 = HISTO(L,6,J)
      D33 = HISTO(L,4,J)
      SIGMA1 = D11*EPS1 + D12*EPS2 + D13*EPS3
      SIGMA2 = D21*EPS1 + D22*EPS2 + D23*EPS3
      SIGMA3 = D31*EPS1 + D32*EPS2 + D33*EPS3
C
      SIGMA1 = DMIN1(PROPER(6),SIGMA1)
      SIGMA2 = DMIN1(PROPER(6),SIGMA2)
      SIGMA3 = DMIN1(PROPER(6),SIGMA3)
      IF (IFLAG .EQ. 4) THEN
          DOT1 = V3(1)*PDIR(1,1) + V3(2)*PDIR(1,2) + V3(3)*PDIR(1,3)
          DOT2 = V3(1)*PDIR(2,1) + V3(2)*PDIR(2,2) + V3(3)*PDIR(2,3)
          DOT3 = V3(1)*PDIR(3,1) + V3(2)*PDIR(3,2) + V3(3)*PDIR(3,3)
          IF (DABS(DOT1) .GE. 0.99) SIGMA1 = 0.0
          IF (DABS(DOT2) .GE. 0.99) SIGMA2 = 0.0
          IF (DABS(DOT3) .GE. 0.99) SIGMA3 = 0.0
      ENDIF
C
      SABS = DMAX1(DMAX1(DABS(SIGMA1),DABS(SIGMA2)),DABS(SIGMA3))
      IF (DABS(SIGMA1) .LT. SABS*1.E-4) SIGMA1 = 0.0
      IF (DABS(SIGMA2) .LT. SABS*1.E-4) SIGMA2 = 0.0
      IF (DABS(SIGMA3) .LT. SABS*1.E-4) SIGMA3 = 0.0
C
      HISTO(L,61,J) = SIGMA1
      HISTO(L,62,J) = SIGMA2
      HISTO(L,63,J) = SIGMA3
C
C
C
  120 CONTINUE
      IFLAG1 = 0
      IFLAG2 = 0
      IFLAG3 = 0
C
C GO TO VECCHIO COLLINS MODEL FOR CRACKED CONCRETE
C
      IF (IFC .EQ. 1) THEN
          IF (CRACK(L,1,J) .GE. 1.0) THEN
              IF (DABS(SIGMA3).GT.PROPER(5)*0.30 .AND. SIGMA3.LT.0.0)
     1            THEN
                  CALL CMPSFT (EPSMAX, EPSMID, EPSMIN, SIGMA1, SIGMA2,
     1                SIGMA3, PROPER, EPS1, EPS2, EPS3, L, J, IOUT,
     2                CONVER, NSTIFF, CHKGUS, TOLER)
                  GO TO 130
              ENDIF
          ENDIF
      ENDIF
C
C         STORE THE TOTAL STRAINS FOR WRITEING OUT LATER
C
      IF (CRACK(L,1,J) .GE. 1.0) THEN
          HISTO(L,38,J) = EPSMAX
          HISTO(L,39,J) = EPSMID
          HISTO(L,40,J) = EPSMIN
          HISTO(L,41,J) = EPSMAX
          HISTO(L,42,J) = EPSMID
```

```
          HISTO(L,43,J) = EPSMIN
      ENDIF
C
C              STRAINING IN THE OPPOSITE DIRECTION
C
      IF (EPS1*HISTO(L,31,J) .LT. 0.0) IFLAG1 = 1
      IF (EPS2*HISTO(L,32,J) .LT. 0.0) IFLAG2 = 1
      IF (EPS3*HISTO(L,33,J) .LT. 0.0) IFLAG3 = 1
      IF (IFLAG1.EQ.1 .OR. IFLAG2.EQ.1 .OR. IFLAG3.EQ.1) THEN
          CALL SUBCON (PDIR, SIGMA1, SIGMA2, SIGMA3, PROPER, L, J,
     1        CONVER, IFLAG1, IFLAG2, IFLAG3, EPS1, EPS2, EPS3, IFC)
          IF (CRACK(L,1,J) .GE. 1.0) GO TO 130
          RETURN
      ENDIF
      IF (DABS(EPS1).GE..999*DABS(HISTO(L,31,J)) .OR. DABS(EPS2).GE..999
     1    *DABS(HISTO(L,32,J)) .OR. DABS(EPS3).GE..999*DABS(HISTO(L,33,J
     2    ))) THEN
          IF (CRACK(L,1,J) .GE. 1.0) THEN
              HISTO(L,10,J) = PDIR(1,1)
              HISTO(L,11,J) = PDIR(1,2)
              HISTO(L,12,J) = PDIR(1,3)
              HISTO(L,13,J) = PDIR(2,1)
              HISTO(L,14,J) = PDIR(2,2)
              HISTO(L,15,J) = PDIR(2,3)
              HISTO(L,16,J) = PDIR(3,1)
              HISTO(L,17,J) = PDIR(3,2)
              HISTO(L,18,J) = PDIR(3,3)
          ENDIF
C
C
C              IF THE POINT WAS UNLOADING OR RELOADING IN
C              THE LAST ITERATION ....
C
C
          IF (HISTO(L,30,J).EQ.2. .OR. HISTO(L,30,J).EQ.3.) THEN
              CALL SUBCON (PDIR, SIGMA1, SIGMA2, SIGMA3, PROPER, L, J,
     1            CONVER, IFLAG1, IFLAG2, IFLAG3, EPS1, EPS2, EPS3, IFC)
              IF (CRACK(L,1,J) .GE. 1.0) GO TO 130
              RETURN
          ENDIF
          IF (HISTO(L,30,J) .EQ. 1.) THEN
C
C
C                        -------------------------
C                        *                       *
C                        *  L  O  A  D  I  N  G  *
C                        *                       *
C                        -------------------------
C
C
C
C             THE POINT WAS PREVIOUSLY LOADING (INCREASE IN STRAIN)
C             AND CONTINUES TO DO SO.
C
C
C             SAVE INTACT CONCRETE'S CURRENT PRINCIPAL STRAINS
C             'EPS1' & 'EPS2' AT THIS POINT.
C
              HISTO(L,22,J) = EPS1
              HISTO(L,23,J) = EPS2
              HISTO(L,24,J) = EPS3
              IF (HISTO(L,29,J) .EQ. 2.) THEN
C
```

```
C
C               COMPARE THE CURRENT PRINCIPAL STRESSES WITH THOSE AT
C               ULTIMATE.IF THERE IS OVERSHOOTING THEN THE MATERIAL
C               IS IN THE POST-ULTIMATE REGION.
C
C
                 SIGM1F = HISTO(L,34,J)
                 SIGM2F = HISTO(L,35,J)
                 SIGM3F = HISTO(L,36,J)
                 STATE = HISTO(L,25,J)
                 IF (STATE .EQ. (-100.)) THEN
                     IF (DABS(SIGMA3) .GT. DABS(SIGM3F)) THEN
                         HISTO(L,10,J) = PDIR(1,1)
                         HISTO(L,11,J) = PDIR(1,2)
                         HISTO(L,12,J) = PDIR(1,3)
                         HISTO(L,13,J) = PDIR(2,1)
                         HISTO(L,14,J) = PDIR(2,2)
                         HISTO(L,15,J) = PDIR(2,3)
                         HISTO(L,16,J) = PDIR(3,1)
                         HISTO(L,17,J) = PDIR(3,2)
                         HISTO(L,18,J) = PDIR(3,3)
                         CALL CON23 (L, J, CONVER, NSTIFF, SIGMA1,
     1                       SIGMA2, SIGMA3, PROPER, IOUT, TOLER,
     2                       CHKGUS, IELEM, IFC)
                         GO TO 130
                     ENDIF
                 ENDIF
                 IF (STATE.EQ.(-110.) .OR. STATE.EQ.(-101) .OR. STATE
     1               .EQ.(-111.)) THEN
                     IF (DABS(SIGMA1).GT.DABS(SIGM1F) .OR. DABS(SIGMA2)
     1                   .GT.DABS(SIGM2F) .OR. DABS(SIGMA3).GT.DABS(
     2                   SIGM3F)) THEN
                         HISTO(L,10,J) = PDIR(1,1)
                         HISTO(L,11,J) = PDIR(1,2)
                         HISTO(L,12,J) = PDIR(1,3)
                         HISTO(L,13,J) = PDIR(2,1)
                         HISTO(L,14,J) = PDIR(2,2)
                         HISTO(L,15,J) = PDIR(2,3)
                         HISTO(L,16,J) = PDIR(3,1)
                         HISTO(L,17,J) = PDIR(3,2)
                         HISTO(L,18,J) = PDIR(3,3)
                         CALL CON23 (L, J, CONVER, NSTIFF, SIGMA1,
     1                       SIGMA2, SIGMA3, PROPER, IOUT, TOLER,
     2                       CHKGUS, IELEM, IFC)
                         GO TO 130
                     ENDIF
                 ENDIF
C
C
C               CHECK TO SEE IF CONVERGENCE IS ACHEIVED AT THE
C               VICINITY OF THE ULTIMATE POINT.
C
C
                 IF (DABS(EPS1).GE..999*DABS(HISTO(L,31,J)) .AND. DABS(
     1               EPS1).LE.1.001*DABS(HISTO(L,31,J))) THEN
                     IF (DABS(EPS2).GE..999*DABS(HISTO(L,32,J)) .AND.
     1                   DABS(EPS2).LE.1.001*DABS(HISTO(L,32,J))) THEN
                         IF (DABS(EPS3).GE..999*DABS(HISTO(L,33,J))
     1                        .AND. DABS(EPS3).LE.1.001*DABS(HISTO(L,33
     2                       ,J))) THEN
C
C
C               THE STRAINS ARE NOT CHANGING.THEREFORE,THE LOADING IS
```

```
C               SUCH THAT THE POINT CONVERGES AT ITS ULTIMATE VALUE.
C
C
                            HISTO(L,10,J) = PDIR(1,1)
                            HISTO(L,11,J) = PDIR(1,2)
                            HISTO(L,12,J) = PDIR(1,3)
                            HISTO(L,13,J) = PDIR(2,1)
                            HISTO(L,14,J) = PDIR(2,2)
                            HISTO(L,15,J) = PDIR(2,3)
                            HISTO(L,16,J) = PDIR(3,1)
                            HISTO(L,17,J) = PDIR(3,2)
                            HISTO(L,18,J) = PDIR(3,3)
                            GO TO 130
                        ENDIF
                    ENDIF
                ENDIF
C
C
C               IF THERE ARE CRACKS AT THIS POINT THEN THE UNLOADING
C               STRAINS ARE THOSE CALCULATED AT THE ONSET OF CRACKING.
C
C
                IF (CRACK(L,1,J) .GE. 1.) THEN
                    HISTO(L,10,J) = PDIR(1,1)
                    HISTO(L,11,J) = PDIR(1,2)
                    HISTO(L,12,J) = PDIR(1,3)
                    HISTO(L,13,J) = PDIR(2,1)
                    HISTO(L,14,J) = PDIR(2,2)
                    HISTO(L,15,J) = PDIR(2,3)
                    HISTO(L,16,J) = PDIR(3,1)
                    HISTO(L,17,J) = PDIR(3,2)
                    HISTO(L,18,J) = PDIR(3,3)
                    CALL CON23 (L, J, CONVER, NSTIFF, SIGMA1, SIGMA2,
     1                  SIGMA3, PROPER, IOUT, TOLER, CHKGUS, IELEM,
     2                  IFC)
                    GO TO 130
                ENDIF
            ENDIF
C
C             CONSIDER POSSIBILITY OF UNLOADING IN NEXT STEPS.
C             THEREFORE,SAVE THE STRAINS FROM WHICH UNLOADING
C             WILL START.
C
C
            HISTO(L,31,J) = EPS1
            HISTO(L,32,J) = EPS2
            HISTO(L,33,J) = EPS3
C
C             IF THE STRESS POINT WAS IN THE PRE-FAILURE PHASE...
C
            IF (HISTO(L,29,J) .EQ. 1.) THEN
                CALL CON12 (SIGMA1, SIGMA2, SIGMA3, L, J, EPS1, EPS2,
     1              EPS3, INCREM, NIT, PROPER, IELEM, NNEL, IOUT,
     2              CONVER, NSTIFF, TOLER, PDIR, CHKGUS, ITYPE, DTSTIF
     3              , IFC, IERROR, IFLG1, IFLG2, IFLG3, ITCRK, ICOUP,
     4              IFLAG)
                GO TO 130
            ENDIF
C
C
C
C             IF THE STRESS POINT,IN PREVIOUS ITERATION,WAS ON THE
C             FAILURE ENVELOPE OR OUTSIDE THE ENVELOPE ........
```

```
C
C
            IF (HISTO(L,29,J).EQ.2. .OR. HISTO(L,29,J).EQ.3.) THEN
                HISTO(L,10,J) = PDIR(1,1)
                HISTO(L,11,J) = PDIR(1,2)
                HISTO(L,12,J) = PDIR(1,3)
                HISTO(L,13,J) = PDIR(2,1)
                HISTO(L,14,J) = PDIR(2,2)
                HISTO(L,15,J) = PDIR(2,3)
                HISTO(L,16,J) = PDIR(3,1)
                HISTO(L,17,J) = PDIR(3,2)
                HISTO(L,18,J) = PDIR(3,3)
                CALL CON23 (L, J, CONVER, NSTIFF, SIGMA1, SIGMA2,
     1              SIGMA3, PROPER, IOUT, TOLER, CHKGUS, IELEM, IFC)
                GO TO 130
            ENDIF
        ENDIF
    ENDIF
C
C
C     COMPARE THE CURRENT PRINCIPAL STRAINS,EPS1 , EPS2 & EPS3.
C     WITH THE PREVIOUS VALUES SAVED IN 'HISTO' TO CHECK
C     UNLOADING OR RELOADING.
C
C
C                     -----------------------------
C                     *                           *
C                     *  UNLOADING  OR  RELOADING  *
C                     *                           *
C                     -----------------------------
C
C
C
    IF (DABS(EPS1).LT..999*DABS(HISTO(L,31,J)) .AND. DABS(EPS2).LT.
     1    .999*DABS(HISTO(L,32,J)) .AND. DABS(EPS3).LT..999*DABS(HISTO(L
     2    ,33,J))) THEN
        IF (CRACK(L,1,J) .EQ. 0.0) THEN
            HISTO(L,10,J) = PDIR(1,1)
            HISTO(L,11,J) = PDIR(1,2)
            HISTO(L,12,J) = PDIR(1,3)
            HISTO(L,13,J) = PDIR(2,1)
            HISTO(L,14,J) = PDIR(2,2)
            HISTO(L,15,J) = PDIR(2,3)
            HISTO(L,16,J) = PDIR(3,1)
            HISTO(L,17,J) = PDIR(3,2)
            HISTO(L,18,J) = PDIR(3,3)
        ENDIF
C
C               DETERMINE STATE OF STRESS AND RATIO OF PRINC. STRESSES
C
        CALL STATUS (SIGMA1, SIGMA2, SIGMA3, STATE, J, PROPER, L, IFC)
        HISTO(L,25,J) = STATE
C
C
C               IF THERE'S OVERSHOOTING IN THE ULTIMATE AND POST-
C               ULTIMATE REGION THEN THE DETECTED UNLOADING-RELOADING
C               CASE MUST BE TREATED WITH CAUTION.IN ORDER TO GET
C               BACK TO STRESS-STRAIN CURVE THE STRAINS CAN TAKE A
C               PATH THAT CAN ACTUALLY BE DETECTED AS UNLOADING WITH
C               NO APPARENT NEED FOR CHANGE IN MATERIAL MODULII.
C               ELIMINATE THE OVERSHOOTING PROBLEM FIRST AND THEN
C               COMPARE STRAINS.
C
```

```
C
        IF (HISTO(L,29,J).EQ.2. .OR. HISTO(L,29,J).EQ.3.) THEN
            SIGM1F = HISTO(L,34,J)
            SIGM2F = HISTO(L,35,J)
            SIGM3F = HISTO(L,36,J)
            STATE = HISTO(L,25,J)
            IF (STATE .EQ. (-100.)) THEN
                IF (DABS(SIGMA3) .GT. DABS(SIGM3F)) THEN
                    CALL CON23 (L, J, CONVER, NSTIFF, SIGMA1, SIGMA2,
     1                  SIGMA3, PROPER, IOUT, TOLER, CHKGUS, IELEM,
     2                  IFC)
                    GO TO 130
                ENDIF
            ENDIF
            IF (STATE.EQ.(-110.) .OR. STATE.EQ.(-101.) .OR. STATE.EQ.(
     1          -111.)) THEN
                IF (DABS(SIGMA1).GT.DABS(SIGM1F) .OR. DABS(SIGMA2).GT.
     1              DABS(SIGM2F) .OR. DABS(SIGMA3).GT.DABS(SIGM3F))
     2              THEN
                    CALL CON23 (L, J, CONVER, NSTIFF, SIGMA1, SIGMA2,
     1                  SIGMA3, PROPER, IOUT, TOLER, CHKGUS, IELEM,
     2                  IFC)
                    GO TO 130
                ENDIF
            ENDIF
C
C
C              IF UNLOADING OCCURS AFTER INITIATION OF CRACKING
C              IN THE PREVIOUS ITERATION THEN THE STRESS POINT IS ON
C              THE PRECRACKING REGION.ALSO OVERSHOOTING BEYOND THE
C              CRACKING STRESS COULD HAVE CAUSED THE FALSE SIGNAL
C              FOR CRACK INITIATION WHILE THE STRESS POINT IS STILL
C              IN THE PRECRACKING STAGE.
C
C
            IF (STATE.EQ.100. .OR. STATE.EQ.110. .OR. STATE.EQ.111.
     1           .OR. STATE.EQ.101) HISTO(L,29,J) = 1.
        ENDIF
        CALL UNRELD(EPS1,EPS2,EPS3,PROPER,CHKGUS,NSTIFF,CONVER,L,J)
    ENDIF
C
C              CALCULATE THE CURRENT STRAIN IN THE DIRECTION OF REBARS
C
  130 CONTINUE
    IF (CHKGUS) THEN
C
C
C             THE MATERIAL LAW AT THE CURRENT GAUSS POINT HAS NOT
C             CONVERGED.UPDATE THE CONSTITUTIVE MATRIX.
C
C
        CALL UPDMAT (MAXCRK, IOUT, PROPER, MAXCK2, L, J, DTSTIF, ICOUP
     1      , IFC, IFLG1, IFLG2, IFLG3, ITCRK, TEMP11, EPSX, EPSY,
     2      EPSZ, GAMAXY, GAMAYZ, GAMAXZ, IELEM)
        RETURN
    ENDIF
    IF (CRACK(L,1,J).GE.1. .AND. EPS1.GT.0.0) THEN
C
C
C              CHECK THE POSSIBILITY OF A NEW CRACK FORMATION.THIS
C              CAN HAPPEN WHEN THE MAX. PRINCIPAL STRAIN OF INTACT
C              CONCRETE EXCEEDS ITS CORRESPONDING VALUE AT THE
C              INITIATION OF THE FIRST CRACK. SUCH CRACKS WERE NOT
```

```
C               OBSERVED IN THE PRESENT STUDY.
C
          CALL ROCRAK (EPS1, SIGMA1, SIGMA2, SIGMA3, PROPER, L, J, IELEM
     1         , NNEL, CHKGUS, NSTIFF, CONVER, IOUT, ITYPE, DTSTIF,
     2         IERROR, IFLG1, IFLG2, IFLG3, ITCRK, ICOUP, IFLAG)
          IF (CHKGUS) GO TO 130
      ENDIF
      RETURN
      END
C**************************************************************
C     INCLUDE(PROCESS)
      SUBROUTINE CTNSTF(ELNUM,INTGPN,ILAYER,IOUT,NLAYRS,ICOUNT)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C**************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C**************************************************************
      INTEGER ELNUM
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON/LPROP/PROPER(25,15)
      COMMON /ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      COMMON/LAYERA/ELLYFO(320,5000)
      DIMENSION ICOUNT(27,15)
C
C     PREPROCESSING TO DETERMINE LAYERS OF CONCRETE THAT ARE
C     TENSION-STIFFENING I.E CONCRETE LAYERS WITH STEEL ADJACENT TO IT.
C
      IF (ICOUNT(INTGPN,ILAYER) .LE. 2) THEN
          IF (ILAYER + 1.LE.NLAYRS .AND. ILAYER.GE.1) THEN
              ICON = (ILAYER-1)*21 + 1
              MAC = ELLYFO(ICON,ELNUM)
              ES = PROPER(1,MAC)
              ICNT1 = ILAYER*21 + 1
              MATRL1 = ELLYFO(ICNT1,ELNUM)
              ICK1 = PROPER(25,MATRL1)
              IF (ICK1 .EQ. 0) THEN
                  EPSY1 = PROPER(5,MATRL1)/PROPER(1,MATRL1)
                  SIGYD1 = PROPER(5,MATRL1)
                  ARATO1 = PROPER(1,MATRL1)/ES
                  ILY1 = ILAYER + 1
                  IF (ILAYER - 1 .GT. 0) THEN
                      ICNT2 = (ILAYER-2)*21 + 1
                      MATRL2 = ELLYFO(ICNT2,ELNUM)
                      ICK2 = PROPER(25,MATRL2)
                      IF (ICK2 .EQ. 0) THEN
                          EPSY2 = PROPER(5,MATRL2)/PROPER(1,MATRL2)
                          SIGYD2 = PROPER(5,MATRL2)
                          ARATO2 = PROPER(1,MATRL2)/ES
                          ILY2 = ILAYER - 1
                          ICNT3 = ICNT1 + 9
                          ICNT4 = ICNT2 + 9
                          AL1 = ELLYFO(ICNT3,ELNUM)
                          AM1 = ELLYFO(ICNT3+1,ELNUM)
                          AN1 = ELLYFO(ICNT3+2,ELNUM)
                          AL2 = ELLYFO(ICNT4,ELNUM)
                          AM2 = ELLYFO(ICNT4+1,ELNUM)
                          AN2 = ELLYFO(ICNT4+2,ELNUM)
```

```
      DOT = AL1*AL2 + AM1*AM2 + AN1*AN2
      IF (DABS(DOT).GE.0.999 .AND. DABS(DOT).LE.
     1    1.001) THEN
          IF (EPSY2 .GT. EPSY1) THEN
              EPSY = EPSY2
              ILY = ILY2
              SIGYLD = SIGYD2
              ARATIO = ARATO2
              AL1 = ELLYFO(ICNT4,ELNUM)
              AM1 = ELLYFO(ICNT4+1,ELNUM)
              AN1 = ELLYFO(ICNT4+2,ELNUM)
              AL2 = ELLYFO(ICNT4+3,ELNUM)
              AM2 = ELLYFO(ICNT4+4,ELNUM)
              AN2 = ELLYFO(ICNT4+5,ELNUM)
              AL3 = ELLYFO(ICNT4+6,ELNUM)
              AM3 = ELLYFO(ICNT4+7,ELNUM)
              AN3 = ELLYFO(ICNT4+8,ELNUM)
          ELSE
              EPSY = EPSY1
              ILY = ILY1
              SIGYLD = SIGYD1
              ARATIO = ARATO1
              AL1 = ELLYFO(ICNT3,ELNUM)
              AM1 = ELLYFO(ICNT3+1,ELNUM)
              AN1 = ELLYFO(ICNT3+2,ELNUM)
              AL2 = ELLYFO(ICNT3+3,ELNUM)
              AM2 = ELLYFO(ICNT3+4,ELNUM)
              AN2 = ELLYFO(ICNT3+5,ELNUM)
              AL3 = ELLYFO(ICNT3+6,ELNUM)
              AM3 = ELLYFO(ICNT3+7,ELNUM)
              AN3 = ELLYFO(ICNT3+8,ELNUM)
          ENDIF
          ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,
     1        ILAYER) + 1
          IPNT = ICOUNT(INTGPN,ILAYER)
          TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
          TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
          TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
          TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
          TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
          IMT = 9*(IPNT-1) + 1
          TENSTF(INTGPN,30+IMT,ILAYER) = AL1
          TENSTF(INTGPN,31+IMT,ILAYER) = AM1
          TENSTF(INTGPN,32+IMT,ILAYER) = AN1
          TENSTF(INTGPN,33+IMT,ILAYER) = AL2
          TENSTF(INTGPN,34+IMT,ILAYER) = AM2
          TENSTF(INTGPN,35+IMT,ILAYER) = AN2
          TENSTF(INTGPN,36+IMT,ILAYER) = AL3
          TENSTF(INTGPN,37+IMT,ILAYER) = AM3
          TENSTF(INTGPN,38+IMT,ILAYER) = AN3
      ELSE IF (ILAYER - 2 .GT. 0) THEN
          ICNT5 = (ILAYER-3)*21 + 1
          MATRL3 = ELLYFO(ICNT5,ELNUM)
          ICK3 = PROPER(25,MATRL3)
          IF (ICK3 .EQ. 0) THEN
              EPSY3 = PROPER(5,MATRL3)/PROPER(1,
     1            MATRL3)
              SIGYD3 = PROPER(5,MATRL3)
              ARATO3 = PROPER(1,MATRL3)/ES
              ILY3 = ILAYER - 2
              ICNT3 = ICNT1 + 9
              ICNT6 = ICNT5 + 9
              AL1 = ELLYFO(ICNT3,ELNUM)
```

```
      AM1 = ELLYFO(ICNT3+1,ELNUM)
      AN1 = ELLYFO(ICNT3+2,ELNUM)
      AL2 = ELLYFO(ICNT6,ELNUM)
      AM2 = ELLYFO(ICNT6+1,ELNUM)
      AN2 = ELLYFO(ICNT6+2,ELNUM)
      DOT = AL1*AL2 + AM1*AM2 + AN1*AN2
      IF (DABS(DOT).GE.0.999 .AND. DABS(DOT)
     1    .LE.1.001) THEN
      IF (EPSY3 .GT. EPSY1) THEN
      EPSY = EPSY3
      ILY = ILY3
      SIGYLD = SIGYD3
      ARATIO = ARATO3
      AL1 = ELLYFO(ICNT6,ELNUM)
      AM1 = ELLYFO(ICNT6+1,ELNUM)
      AN1 = ELLYFO(ICNT6+2,ELNUM)
      AL2 = ELLYFO(ICNT6+3,ELNUM)
      AM2 = ELLYFO(ICNT6+4,ELNUM)
      AN2 = ELLYFO(ICNT6+5,ELNUM)
      AL3 = ELLYFO(ICNT6+6,ELNUM)
      AM3 = ELLYFO(ICNT6+7,ELNUM)
      AN3 = ELLYFO(ICNT6+8,ELNUM)
      ELSE
      EPSY = EPSY1
      ILY = ILY1
      SIGYLD = SIGYD1
      ARATIO = ARATO1
      AL1 = ELLYFO(ICNT3,ELNUM)
      AM1 = ELLYFO(ICNT3+1,ELNUM)
      AN1 = ELLYFO(ICNT3+2,ELNUM)
      AL2 = ELLYFO(ICNT3+3,ELNUM)
      AM2 = ELLYFO(ICNT3+4,ELNUM)
      AN2 = ELLYFO(ICNT3+5,ELNUM)
      AL3 = ELLYFO(ICNT3+6,ELNUM)
      AM3 = ELLYFO(ICNT3+7,ELNUM)
      AN3 = ELLYFO(ICNT3+8,ELNUM)
      ENDIF
      ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,
     1    ILAYER) + 1
      IPNT = ICOUNT(INTGPN,ILAYER)
      TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
      TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
      TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
      TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
      TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
      IMT = 9*(IPNT-1) + 1
      TENSTF(INTGPN,30+IMT,ILAYER) = AL1
      TENSTF(INTGPN,31+IMT,ILAYER) = AM1
      TENSTF(INTGPN,32+IMT,ILAYER) = AN1
      TENSTF(INTGPN,33+IMT,ILAYER) = AL2
      TENSTF(INTGPN,34+IMT,ILAYER) = AM2
      TENSTF(INTGPN,35+IMT,ILAYER) = AN2
      TENSTF(INTGPN,36+IMT,ILAYER) = AL3
      TENSTF(INTGPN,37+IMT,ILAYER) = AM3
      TENSTF(INTGPN,38+IMT,ILAYER) = AN3
      ELSE
      EPSY = EPSY1
      ILY = ILY1
      SIGYLD = SIGYD1
      ARATIO = ARATO1
      AL1 = ELLYFO(ICNT3,ELNUM)
      AM1 = ELLYFO(ICNT3+1,ELNUM)
      AN1 = ELLYFO(ICNT3+2,ELNUM)
```

```
                    AL2 = ELLYFO(ICNT3+3,ELNUM)
                    AM2 = ELLYFO(ICNT3+4,ELNUM)
                    AN2 = ELLYFO(ICNT3+5,ELNUM)
                    AL3 = ELLYFO(ICNT3+6,ELNUM)
                    AM3 = ELLYFO(ICNT3+7,ELNUM)
                    AN3 = ELLYFO(ICNT3+8,ELNUM)
                    ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,
     1                  ILAYER) + 1
                    IPNT = ICOUNT(INTGPN,ILAYER)
                    TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
                    TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
                    TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
                    TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
                    TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
                    IMT = 9*(IPNT-1) + 1
                    TENSTF(INTGPN,30+IMT,ILAYER) = AL1
                    TENSTF(INTGPN,31+IMT,ILAYER) = AM1
                    TENSTF(INTGPN,32+IMT,ILAYER) = AN1
                    TENSTF(INTGPN,33+IMT,ILAYER) = AL2
                    TENSTF(INTGPN,34+IMT,ILAYER) = AM2
                    TENSTF(INTGPN,35+IMT,ILAYER) = AN2
                    TENSTF(INTGPN,36+IMT,ILAYER) = AL3


                    TENSTF(INTGPN,37+IMT,ILAYER) = AM3
                    TENSTF(INTGPN,38+IMT,ILAYER) = AN3
                    ENDIF
                ELSE
                    ICNT3 = ICNT1 + 9
                    EPSY = EPSY1
                    ILY = ILY1
                    SIGYLD = SIGYD1
                    ARATIO = ARATO1
                    AL1 = ELLYFO(ICNT3,ELNUM)
                    AM1 = ELLYFO(ICNT3+1,ELNUM)
                    AN1 = ELLYFO(ICNT3+2,ELNUM)
                    AL2 = ELLYFO(ICNT3+3,ELNUM)
                    AM2 = ELLYFO(ICNT3+4,ELNUM)
                    AN2 = ELLYFO(ICNT3+5,ELNUM)
                    AL3 = ELLYFO(ICNT3+6,ELNUM)
                    AM3 = ELLYFO(ICNT3+7,ELNUM)
                    AN3 = ELLYFO(ICNT3+8,ELNUM)
                    ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,
     1                  ILAYER) + 1
                    IPNT = ICOUNT(INTGPN,ILAYER)
                    TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
                    TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
                    TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
                    TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
                    TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
                    IMT = 9*(IPNT-1) + 1
                    TENSTF(INTGPN,30+IMT,ILAYER) = AL1
                    TENSTF(INTGPN,31+IMT,ILAYER) = AM1
                    TENSTF(INTGPN,32+IMT,ILAYER) = AN1
                    TENSTF(INTGPN,33+IMT,ILAYER) = AL2
                    TENSTF(INTGPN,34+IMT,ILAYER) = AM2
                    TENSTF(INTGPN,35+IMT,ILAYER) = AN2
                    TENSTF(INTGPN,36+IMT,ILAYER) = AL3
                    TENSTF(INTGPN,37+IMT,ILAYER) = AM3
                    TENSTF(INTGPN,38+IMT,ILAYER) = AN3
                ENDIF
            ELSE
                ICNT3 = ICNT1 + 9
                EPSY = EPSY1
```

```
                  ILY = ILY1
                  SIGYLD = SIGYD1
                  ARATIO = ARATO1
                  AL1 = ELLYFO(ICNT3,ELNUM)
                  AM1 = ELLYFO(ICNT3+1,ELNUM)
                  AN1 = ELLYFO(ICNT3+2,ELNUM)
                  AL2 = ELLYFO(ICNT3+3,ELNUM)
                  AM2 = ELLYFO(ICNT3+4,ELNUM)
                  AN2 = ELLYFO(ICNT3+5,ELNUM)
                  AL3 = ELLYFO(ICNT3+6,ELNUM)
                  AM3 = ELLYFO(ICNT3+7,ELNUM)
                  AN3 = ELLYFO(ICNT3+8,ELNUM)
                  ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,
     1                ILAYER) + 1
                  IPNT = ICOUNT(INTGPN,ILAYER)
                  TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
                  TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
                  TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
                  TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
                  TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
                  IMT = 9*(IPNT-1) + 1
                  TENSTF(INTGPN,30+IMT,ILAYER) = AL1
                  TENSTF(INTGPN,31+IMT,ILAYER) = AM1
                  TENSTF(INTGPN,32+IMT,ILAYER) = AN1
                  TENSTF(INTGPN,33+IMT,ILAYER) = AL2
                  TENSTF(INTGPN,34+IMT,ILAYER) = AM2
                  TENSTF(INTGPN,35+IMT,ILAYER) = AN2
                  TENSTF(INTGPN,36+IMT,ILAYER) = AL3
                  TENSTF(INTGPN,37+IMT,ILAYER) = AM3
                  TENSTF(INTGPN,38+IMT,ILAYER) = AN3
              ENDIF
          ELSE
              ICNT3 = ICNT1 + 9
              EPSY = EPSY1
              ILY = ILY1
              SIGYLD = SIGYD1
              ARATIO = ARATO1
              AL1 = ELLYFO(ICNT3,ELNUM)
              AM1 = ELLYFO(ICNT3+1,ELNUM)
              AN1 = ELLYFO(ICNT3+2,ELNUM)
              AL2 = ELLYFO(ICNT3+3,ELNUM)
              AM2 = ELLYFO(ICNT3+4,ELNUM)
              AN2 = ELLYFO(ICNT3+5,ELNUM)
              AL3 = ELLYFO(ICNT3+6,ELNUM)
              AM3 = ELLYFO(ICNT3+7,ELNUM)
              AN3 = ELLYFO(ICNT3+8,ELNUM)
              ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,ILAYER)
     1             + 1
              IPNT = ICOUNT(INTGPN,ILAYER)
              TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
              TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
              TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
              TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
              TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
              IMT = 9*(IPNT-1) + 1
              TENSTF(INTGPN,30+IMT,ILAYER) = AL1
              TENSTF(INTGPN,31+IMT,ILAYER) = AM1
              TENSTF(INTGPN,32+IMT,ILAYER) = AN1
              TENSTF(INTGPN,33+IMT,ILAYER) = AL2
              TENSTF(INTGPN,34+IMT,ILAYER) = AM2
              TENSTF(INTGPN,35+IMT,ILAYER) = AN2
              TENSTF(INTGPN,36+IMT,ILAYER) = AL3
              TENSTF(INTGPN,37+IMT,ILAYER) = AM3
```

```
                  TENSTF(INTGPN,38+IMT,ILAYER) = AN3
              ENDIF
          ELSE
              ICNT3 = ICNT1 + 9
              EPSY = EPSY1
              ILY = ILY1
              SIGYLD = SIGYD1
              ARATIO = ARATO1
              AL1 = ELLYFO(ICNT3,ELNUM)
              AM1 = ELLYFO(ICNT3+1,ELNUM)
              AN1 = ELLYFO(ICNT3+2,ELNUM)
              AL2 = ELLYFO(ICNT3+3,ELNUM)
              AM2 = ELLYFO(ICNT3+4,ELNUM)
              AN2 = ELLYFO(ICNT3+5,ELNUM)
              AL3 = ELLYFO(ICNT3+6,ELNUM)
              AM3 = ELLYFO(ICNT3+7,ELNUM)
              AN3 = ELLYFO(ICNT3+8,ELNUM)
              ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,ILAYER) + 1
              IPNT = ICOUNT(INTGPN,ILAYER)
              TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
              TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
              TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
              TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
              TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
              IMT = 9*(IPNT-1) + 1
              TENSTF(INTGPN,30+IMT,ILAYER) = AL1
              TENSTF(INTGPN,31+IMT,ILAYER) = AM1
              TENSTF(INTGPN,32+IMT,ILAYER) = AN1
              TENSTF(INTGPN,33+IMT,ILAYER) = AL2
              TENSTF(INTGPN,34+IMT,ILAYER) = AM2
              TENSTF(INTGPN,35+IMT,ILAYER) = AN2
              TENSTF(INTGPN,36+IMT,ILAYER) = AL3
              TENSTF(INTGPN,37+IMT,ILAYER) = AM3
              TENSTF(INTGPN,38+IMT,ILAYER) = AN3
          ENDIF
      ENDIF
  ENDIF
ENDIF
C


C
C
      IF (ICOUNT(INTGPN,ILAYER) .LE. 2) THEN
          IF (ILAYER + 2.LE.NLAYRS .AND. ILAYER.GE.1) THEN
              ICON = (ILAYER-1)*21 + 1
              MAC = ELLYFO(ICON,ELNUM)
              ES = PROPER(1,MAC)
              ICNT1 = (ILAYER+1)*21 + 1
              MATRL1 = ELLYFO(ICNT1,ELNUM)
              ICK1 = PROPER(25,MATRL1)
              IF (ICK1 .EQ. 0) THEN
                  ILY1 = ILAYER + 2
                  EPSY1 = PROPER(5,MATRL1)/PROPER(1,MATRL1)
                  SIGYD1 = PROPER(5,MATRL1)
                  ARATO1 = PROPER(1,MATRL1)/ES
                  ICNT2 = ILAYER*21 + 1
                  MATRL2 = ELLYFO(ICNT2,ELNUM)
                  ICK2 = PROPER(25,MATRL2)
                  IF (ICK2 .EQ. 0) THEN
                      IF (ILAYER - 1 .GT. 0) THEN
                          ICNT3 = (ILAYER-2)*21 + 1
                          MATRL3 = ELLYFO(ICNT3,ELNUM)
                          ICK3 = PROPER(25,MATRL3)
```

```
      IF (ICK3 .EQ. 0) THEN
          ILY3 = ILAYER - 1
          EPSY3 = PROPER(5,MATRL3)/PROPER(1,MATRL3)
          SIGYD3 = PROPER(5,MATRL3)
          ARATO3 = PROPER(1,MATRL3)/ES
          ICNT4 = ICNT1 + 9
          ICNT5 = ICNT3 + 9
          AL1 = ELLYFO(ICNT4,ELNUM)
          AM1 = ELLYFO(ICNT4+1,ELNUM)
          AN1 = ELLYFO(ICNT4+2,ELNUM)
          AL2 = ELLYFO(ICNT5,ELNUM)
          AM2 = ELLYFO(ICNT5+1,ELNUM)
          AN2 = ELLYFO(ICNT5+2,ELNUM)
          DOT = AL1*AL2 + AM1*AM2 + AN1*AN2
          IF (DABS(DOT).GE.0.999 .AND. DABS(DOT).LE.
     1        1.001) THEN
              IF (EPSY3 .GT. EPSY1) THEN
              EPSY = EPSY3
              ILY = ILY3
              SIGYLD = SIGYD3
              ARATIO = ARATO3
              AL1 = ELLYFO(ICNT5,ELNUM)
              AM1 = ELLYFO(ICNT5+1,ELNUM)
              AN1 = ELLYFO(ICNT5+2,ELNUM)
              AL2 = ELLYFO(ICNT5+3,ELNUM)
              AM2 = ELLYFO(ICNT5+4,ELNUM)
              AN2 = ELLYFO(ICNT5+5,ELNUM)
              AL3 = ELLYFO(ICNT5+6,ELNUM)
              AM3 = ELLYFO(ICNT5+7,ELNUM)
              AN3 = ELLYFO(ICNT5+8,ELNUM)
              ELSE
              EPSY = EPSY1
              ILY = ILY1
              SIGYLD = SIGYD1
              ARATIO = ARATO1
              AL1 = ELLYFO(ICNT4,ELNUM)
              AM1 = ELLYFO(ICNT4+1,ELNUM)
              AN1 = ELLYFO(ICNT4+2,ELNUM)
              AL2 = ELLYFO(ICNT4+3,ELNUM)
              AM2 = ELLYFO(ICNT4+4,ELNUM)
              AN2 = ELLYFO(ICNT4+5,ELNUM)
              AL3 = ELLYFO(ICNT4+6,ELNUM)
              AM3 = ELLYFO(ICNT4+7,ELNUM)
              AN3 = ELLYFO(ICNT4+8,ELNUM)
              ENDIF
              ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,
     1            ILAYER) + 1
              IPNT = ICOUNT(INTGPN,ILAYER)
              TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
              TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
              TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
              TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
              TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
              IMT = 9*(IPNT-1) + 1
              TENSTF(INTGPN,30+IMT,ILAYER) = AL1
              TENSTF(INTGPN,31+IMT,ILAYER) = AM1
              TENSTF(INTGPN,32+IMT,ILAYER) = AN1
              TENSTF(INTGPN,33+IMT,ILAYER) = AL2
              TENSTF(INTGPN,34+IMT,ILAYER) = AM2
              TENSTF(INTGPN,35+IMT,ILAYER) = AN2
              TENSTF(INTGPN,36+IMT,ILAYER) = AL3
              TENSTF(INTGPN,37+IMT,ILAYER) = AM3
              TENSTF(INTGPN,38+IMT,ILAYER) = AN3
```

```
            ELSE IF (ILAYER - 2 .GT. 0) THEN
                ICNT6 = (ILAYER-3)*21 + 1
                MATRL4 = ELLYFO(ICNT6,ELNUM)
                ICK4 = PROPER(25,MATRL4)
                IF (ICK4 .EQ. 0) THEN
                ILY4 = ILAYER - 2
                EPSY4 = PROPER(5,MATRL4)/PROPER(1,
     1              MATRL4)
                ARATO4 = PROPER(1,MATRL4)/ES
                SIGYD4 = PROPER(5,MATRL4)
                ICNT3 = ICNT1 + 9
                ICNT6 = ICNT5 + 9
                AL1 = ELLYFO(ICNT3,ELNUM)
                AM1 = ELLYFO(ICNT3+1,ELNUM)
                AN1 = ELLYFO(ICNT3+2,ELNUM)
                AL2 = ELLYFO(ICNT6,ELNUM)
                AM2 = ELLYFO(ICNT6+1,ELNUM)
                AN2 = ELLYFO(ICNT6+2,ELNUM)
                DOT = AL1*AL2 + AM1*AM2 + AN1*AN2
                IF (DABS(DOT).GE.0.999 .AND. DABS(DOT)
     1              .LE.1.001) THEN
                IF (EPSY4 .GT. EPSY1) THEN
                EPSY = EPSY4
                ILY = ILY4
                SIGYLD = SIGYD4
                ARATIO = ARATO4
                AL1 = ELLYFO(ICNT6,ELNUM)
                AM1 = ELLYFO(ICNT6+1,ELNUM)
                AN1 = ELLYFO(ICNT6+2,ELNUM)
                AL2 = ELLYFO(ICNT6+3,ELNUM)
                AM2 = ELLYFO(ICNT6+4,ELNUM)
                AN2 = ELLYFO(ICNT6+5,ELNUM)
                AL3 = ELLYFO(ICNT6+6,ELNUM)
                AM3 = ELLYFO(ICNT6+7,ELNUM)
                AN3 = ELLYFO(ICNT6+8,ELNUM)
                ELSE
                EPSY = EPSY1
                ILY = ILY1
                ARATIO = ARATO1
                SIGYLD = SIGYD1
                AL1 = ELLYFO(ICNT4,ELNUM)
                AM1 = ELLYFO(ICNT4+1,ELNUM)
                AN1 = ELLYFO(ICNT4+2,ELNUM)
                AL2 = ELLYFO(ICNT4+3,ELNUM)
                AM2 = ELLYFO(ICNT4+4,ELNUM)
                AN2 = ELLYFO(ICNT4+5,ELNUM)
                AL3 = ELLYFO(ICNT4+6,ELNUM)
                AM3 = ELLYFO(ICNT4+7,ELNUM)
                AN3 = ELLYFO(ICNT4+8,ELNUM)
                ENDIF
                ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,
     1              ILAYER) + 1
                IPNT = ICOUNT(INTGPN,ILAYER)
                TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
                TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
                TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
                TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
                TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
                IMT = 9*(IPNT-1) + 1
                TENSTF(INTGPN,30+IMT,ILAYER) = AL1
                TENSTF(INTGPN,31+IMT,ILAYER) = AM1
                TENSTF(INTGPN,32+IMT,ILAYER) = AN1
                TENSTF(INTGPN,33+IMT,ILAYER) = AL2
```

```
      TENSTF(INTGPN,34+IMT,ILAYER) = AM2
      TENSTF(INTGPN,35+IMT,ILAYER) = AN2
      TENSTF(INTGPN,36+IMT,ILAYER) = AL3
      TENSTF(INTGPN,37+IMT,ILAYER) = AM3
      TENSTF(INTGPN,38+IMT,ILAYER) = AN3
      ELSE
      ICNT3 = ICNT1 + 9
      EPSY = EPSY1
      ILY = ILY1
      SIGYLD = SIGYD1
      ARATIO = ARATO1
      AL1 = ELLYFO(ICNT3,ELNUM)
      AM1 = ELLYFO(ICNT3+1,ELNUM)
      AN1 = ELLYFO(ICNT3+2,ELNUM)
      AL2 = ELLYFO(ICNT3+3,ELNUM)
      AM2 = ELLYFO(ICNT3+4,ELNUM)
      AN2 = ELLYFO(ICNT3+5,ELNUM)
      AL3 = ELLYFO(ICNT3+6,ELNUM)
      AM3 = ELLYFO(ICNT3+7,ELNUM)
      AN3 = ELLYFO(ICNT3+8,ELNUM)
      ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,
     1    ILAYER) + 1
      IPNT = ICOUNT(INTGPN,ILAYER)
      TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
      TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
      TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
      TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
      TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
      IMT = 9*(IPNT-1) + 1
      TENSTF(INTGPN,30+IMT,ILAYER) = AL1
      TENSTF(INTGPN,31+IMT,ILAYER) = AM1
      TENSTF(INTGPN,32+IMT,ILAYER) = AN1
      TENSTF(INTGPN,33+IMT,ILAYER) = AL2
      TENSTF(INTGPN,34+IMT,ILAYER) = AM2
      TENSTF(INTGPN,35+IMT,ILAYER) = AN2
      TENSTF(INTGPN,36+IMT,ILAYER) = AL3
      TENSTF(INTGPN,37+IMT,ILAYER) = AM3
      TENSTF(INTGPN,38+IMT,ILAYER) = AN3
      ENDIF
      ELSE
      EPSY = EPSY1
      ILY = ILY1
      SIGYLD = SIGYD1
      ARATIO = ARATO1
      ICNT3 = ICNT1 + 9
      AL1 = ELLYFO(ICNT3,ELNUM)
      AM1 = ELLYFO(ICNT3+1,ELNUM)
      AN1 = ELLYFO(ICNT3+2,ELNUM)
      AL2 = ELLYFO(ICNT3+3,ELNUM)
      AM2 = ELLYFO(ICNT3+4,ELNUM)
      AN2 = ELLYFO(ICNT3+5,ELNUM)
      AL3 = ELLYFO(ICNT3+6,ELNUM)
      AM3 = ELLYFO(ICNT3+7,ELNUM)
      AN3 = ELLYFO(ICNT3+8,ELNUM)
      ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,
     1    ILAYER) + 1
      IPNT = ICOUNT(INTGPN,ILAYER)
      TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
      TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
      TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
      TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
      TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
      IMT = 9*(IPNT-1) + 1
```

```
                    TENSTF(INTGPN,30+IMT,ILAYER) = AL1
                    TENSTF(INTGPN,31+IMT,ILAYER) = AM1
                    TENSTF(INTGPN,32+IMT,ILAYER) = AN1
                    TENSTF(INTGPN,33+IMT,ILAYER) = AL2
                    TENSTF(INTGPN,34+IMT,ILAYER) = AM2
                    TENSTF(INTGPN,35+IMT,ILAYER) = AN2
                    TENSTF(INTGPN,36+IMT,ILAYER) = AL3
                    TENSTF(INTGPN,37+IMT,ILAYER) = AM3
                    TENSTF(INTGPN,38+IMT,ILAYER) = AN3
                    ENDIF
                ELSE
                    EPSY = EPSY1
                    ILY = ILY1
                    SIGYLD = SIGYD1
                    ARATIO = ARATO1
                    ICNT3 = ICNT1 + 9
                    AL1 = ELLYFO(ICNT3,ELNUM)
                    AM1 = ELLYFO(ICNT3+1,ELNUM)
                    AN1 = ELLYFO(ICNT3+2,ELNUM)
                    AL2 = ELLYFO(ICNT3+3,ELNUM)
                    AM2 = ELLYFO(ICNT3+4,ELNUM)
                    AN2 = ELLYFO(ICNT3+5,ELNUM)
                    AL3 = ELLYFO(ICNT3+6,ELNUM)
                    AM3 = ELLYFO(ICNT3+7,ELNUM)
                    AN3 = ELLYFO(ICNT3+8,ELNUM)
                    ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,
     1                  ILAYER) + 1
                    IPNT = ICOUNT(INTGPN,ILAYER)
                    TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
                    TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
                    TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
                    TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
                    TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
                    IMT = 9*(IPNT-1) + 1
                    TENSTF(INTGPN,30+IMT,ILAYER) = AL1
                    TENSTF(INTGPN,31+IMT,ILAYER) = AM1
                    TENSTF(INTGPN,32+IMT,ILAYER) = AN1
                    TENSTF(INTGPN,33+IMT,ILAYER) = AL2
                    TENSTF(INTGPN,34+IMT,ILAYER) = AM2
                    TENSTF(INTGPN,35+IMT,ILAYER) = AN2
                    TENSTF(INTGPN,36+IMT,ILAYER) = AL3
                    TENSTF(INTGPN,37+IMT,ILAYER) = AM3
                    TENSTF(INTGPN,38+IMT,ILAYER) = AN3
                ENDIF
            ELSE
                EPSY = EPSY1

                ILY = ILY1
                SIGYLD = SIGYD1
                ARATIO = ARATO1
                ICNT3 = ICNT1 + 9
                AL1 = ELLYFO(ICNT3,ELNUM)
                AM1 = ELLYFO(ICNT3+1,ELNUM)
                AN1 = ELLYFO(ICNT3+2,ELNUM)
                AL2 = ELLYFO(ICNT3+3,ELNUM)
                AM2 = ELLYFO(ICNT3+4,ELNUM)
                AN2 = ELLYFO(ICNT3+5,ELNUM)
                AL3 = ELLYFO(ICNT3+6,ELNUM)
                AM3 = ELLYFO(ICNT3+7,ELNUM)
                AN3 = ELLYFO(ICNT3+8,ELNUM)
                ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,
     1              ILAYER) + 1
                IPNT = ICOUNT(INTGPN,ILAYER)
```

```
                        TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
                        TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
                        TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
                        TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
                        TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
                        IMT = 9*(IPNT-1) + 1
                        TENSTF(INTGPN,30+IMT,ILAYER) = AL1
                        TENSTF(INTGPN,31+IMT,ILAYER) = AM1
                        TENSTF(INTGPN,32+IMT,ILAYER) = AN1
                        TENSTF(INTGPN,33+IMT,ILAYER) = AL2
                        TENSTF(INTGPN,34+IMT,ILAYER) = AM2
                        TENSTF(INTGPN,35+IMT,ILAYER) = AN2
                        TENSTF(INTGPN,36+IMT,ILAYER) = AL3
                        TENSTF(INTGPN,37+IMT,ILAYER) = AM3
                        TENSTF(INTGPN,38+IMT,ILAYER) = AN3
                    ENDIF
                ELSE
                    EPSY = EPSY1
                    ILY = ILY1
                    SIGYLD = SIGYD1
                    ARATIO = ARATO1
                    ICNT3 = ICNT1 + 9
                    AL1 = ELLYFO(ICNT3,ELNUM)
                    AM1 = ELLYFO(ICNT3+1,ELNUM)
                    AN1 = ELLYFO(ICNT3+2,ELNUM)
                    AL2 = ELLYFO(ICNT3+3,ELNUM)
                    AM2 = ELLYFO(ICNT3+4,ELNUM)
                    AN2 = ELLYFO(ICNT3+5,ELNUM)
                    AL3 = ELLYFO(ICNT3+6,ELNUM)
                    AM3 = ELLYFO(ICNT3+7,ELNUM)
                    AN3 = ELLYFO(ICNT3+8,ELNUM)
                    ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,ILAYER)
     1                   + 1
                    IPNT = ICOUNT(INTGPN,ILAYER)
                    TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
                    TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
                    TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
                    TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
                    TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
                    IMT = 9*(IPNT-1) + 1
                    TENSTF(INTGPN,30+IMT,ILAYER) = AL1
                    TENSTF(INTGPN,31+IMT,ILAYER) = AM1
                    TENSTF(INTGPN,32+IMT,ILAYER) = AN1
                    TENSTF(INTGPN,33+IMT,ILAYER) = AL2
                    TENSTF(INTGPN,34+IMT,ILAYER) = AM2
                    TENSTF(INTGPN,35+IMT,ILAYER) = AN2
                    TENSTF(INTGPN,36+IMT,ILAYER) = AL3
                    TENSTF(INTGPN,37+IMT,ILAYER) = AM3
                    TENSTF(INTGPN,38+IMT,ILAYER) = AN3
                ENDIF
            ENDIF
        ENDIF
    ENDIF
      ENDIF
C
      IF (ICOUNT(INTGPN,ILAYER) .LE. 2) THEN
          IF (ILAYER - 1.GE.1 .AND. ILAYER.LE.NLAYRS) THEN
              ICON = (ILAYER-1)*21 + 1
              MAC = ELLYFO(ICON,ELNUM)
              ES = PROPER(1,MAC)
              ICNT1 = (ILAYER-2)*21 + 1
              MATRL1 = ELLYFO(ICNT1,ELNUH)
              ICK1 = PROPER(25,MATRL1)
```

```
        IF (ICK1 .EQ. 0) THEN
            EPSY1 = PROPER(5,MATRL1)/PROPER(1,MATRL1)
            SIGYD1 = PROPER(5,MATRL1)
            ARATO1 = PROPER(1,MATRL1)/ES
            ILY1 = ILAYER - 1
            IF (ILAYER + 1 .LE. NLAYRS) THEN
                ICNT2 = ILAYER*21 + 1
                MATRL2 = ELLYFO(ICNT2,ELNUM)
                ICK2 = PROPER(25,MATRL2)
                IF (ICK2 .EQ. 0) THEN
                    EPSY2 = PROPER(5,MATRL2)/PROPER(1,MATRL2)
                    SIGYD2 = PROPER(5,MATRL2)
                    ARATO2 = PROPER(1,MATRL2)/ES
                    ILY2 = ILAYER + 1
                    ICNT3 = ICNT1 + 9
                    ICNT4 = ICNT2 + 9
                    AL1 = ELLYFO(ICNT3,ELNUM)
                    AM1 = ELLYFO(ICNT3+1,ELNUM)
                    AN1 = ELLYFO(ICNT3+2,ELNUM)
                    AL2 = ELLYFO(ICNT4,ELNUM)
                    AM2 = ELLYFO(ICNT4+1,ELNUM)
                    AN2 = ELLYFO(ICNT4+2,ELNUM)
                    DOT = AL1*AL2 + AM1*AM2 + AN1*AN2
                    IF (DABS(DOT).GE.0.999 .AND. DABS(DOT).LE.
1                       1.001) THEN
                        IF (EPSY2 .GT. EPSY1) THEN
                            EPSY = EPSY2
                            ILY = ILY2
                            SIGYLD = SIGYD2
                            ARATIO = ARATO2
                            AL1 = ELLYFO(ICNT4,ELNUM)
                            AM1 = ELLYFO(ICNT4+1,ELNUM)
                            AN1 = ELLYFO(ICNT4+2,ELNUM)
                            AL2 = ELLYFO(ICNT4+3,ELNUM)
                            AM2 = ELLYFO(ICNT4+4,ELNUM)
                            AN2 = ELLYFO(ICNT4+5,ELNUM)
                            AL3 = ELLYFO(ICNT4+6,ELNUM)
                            AM3 = ELLYFO(ICNT4+7,ELNUM)
                            AN3 = ELLYFO(ICNT4+8,ELNUM)
                        ELSE
                            EPSY = EPSY1
                            ILY = ILY1
                            SIGYLD = SIGYD1
                            ARATIO = ARATO1
                            AL1 = ELLYFO(ICNT3,ELNUM)
                            AM1 = ELLYFO(ICNT3+1,ELNUM)
                            AN1 = ELLYFO(ICNT3+2,ELNUM)
                            AL2 = ELLYFO(ICNT3+3,ELNUM)
                            AM2 = ELLYFO(ICNT3+4,ELNUM)
                            AN2 = ELLYFO(ICNT3+5,ELNUM)
                            AL3 = ELLYFO(ICNT3+6,ELNUM)
                            AM3 = ELLYFO(ICNT3+7,ELNUM)
                            AN3 = ELLYFO(ICNT3+8,ELNUM)
                        ENDIF
                        ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,
1                           ILAYER) + 1
                        IPNT = ICOUNT(INTGPN,ILAYER)
                        TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
                        TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
                        TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
                        TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
                        TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
                        IMT = 9*(IPNT-1) + 1
```

```
          TENSTF(INTGPN,30+IMT,ILAYER) = AL1
          TENSTF(INTGPN,31+IMT,ILAYER) = AM1
          TENSTF(INTGPN,32+IMT,ILAYER) = AN1
          TENSTF(INTGPN,33+IMT,ILAYER) = AL2
          TENSTF(INTGPN,34+IMT,ILAYER) = AM2
          TENSTF(INTGPN,35+IMT,ILAYER) = AN2
          TENSTF(INTGPN,36+IMT,ILAYER) = AL3
          TENSTF(INTGPN,37+IMT,ILAYER) = AM3
          TENSTF(INTGPN,38+IMT,ILAYER) = AN3
      ELSE IF (ILAYER + 2 .LE. NLAYRS) THEN
          ICNT5 = (ILAYER+1)*21 + 1
          MATRL3 = ELLYFO(ICNT5,ELNUM)
          ICK3 = PROPER(25,MATRL3)
          IF (ICK3 .EQ. 0) THEN
              EPSY3 = PROPER(5,MATRL3)/PROPER(1,
     1            MATRL3)
              SIGYD3 = PROPER(5,MATRL3)
              ARAT03 = PROPER(1,MATRL3)/ES
              ILY3 = ILAYER + 2
              ICNT3 = ICNT1 + 9
              ICNT6 = ICNT5 + 9
              AL1 = ELLYFO(ICNT3,ELNUM)
              AM1 = ELLYFO(ICNT3+1,ELNUM)
              AN1 = ELLYFO(ICNT3+2,ELNUM)
              AL2 = ELLYFO(ICNT6,ELNUM)
              AM2 = ELLYFO(ICNT6+1,ELNUM)
              AN2 = ELLYFO(ICNT6+2,ELNUM)
              DOT = AL1*AL2 + AM1*AM2 + AN1*AN2
              IF (DABS(DOT).GE.0.999 .AND. DABS(DOT)
     1            .LE.1.001) THEN
              IF (EPSY3 .GT. EPSY1) THEN
              EPSY = EPSY3
              ILY = ILY3
              SIGYLD = SIGYD3
              ARATIO = ARAT03
              AL1 = ELLYFO(ICNT6,ELNUM)
              AM1 = ELLYFO(ICNT6+1,ELNUM)
              AN1 = ELLYFO(ICNT6+2,ELNUM)
              AL2 = ELLYFO(ICNT6+3,ELNUM)
              AM2 = ELLYFO(ICNT6+4,ELNUM)
              AN2 = ELLYFO(ICNT6+5,ELNUM)
              AL3 = ELLYFO(ICNT6+6,ELNUM)
              AM3 = ELLYFO(ICNT6+7,ELNUM)
              AN3 = ELLYFO(ICNT6+8,ELNUM)
              ELSE
              EPSY = EPSY1
              ILY = ILY1
              SIGYLD = SIGYD1
              ARATIO = ARAT01
              AL1 = ELLYFO(ICNT3,ELNUM)
              AM1 = ELLYFO(ICNT3+1,ELNUM)
              AN1 = ELLYFO(ICNT3+2,ELNUM)
              AL2 = ELLYFO(ICNT3+3,ELNUM)
              AM2 = ELLYFO(ICNT3+4,ELNUM)
              AN2 = ELLYFO(ICNT3+5,ELNUM)
              AL3 = ELLYFO(ICNT3+6,ELNUM)
              AM3 = ELLYFO(ICNT3+7,ELNUM)
              AN3 = ELLYFO(ICNT3+8,ELNUM)


              ENDIF
              ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,
     1            ILAYER) + 1
              IPNT = ICOUNT(INTGPN,ILAYER)
```

```
                  TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
                  TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
                  TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
                  TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
                  TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
                  IMT = 9*(IPNT-1) + 1
                  TENSTF(INTGPN,30+IMT,ILAYER) = AL1
                  TENSTF(INTGPN,31+IMT,ILAYER) = AM1
                  TENSTF(INTGPN,32+IMT,ILAYER) = AN1
                  TENSTF(INTGPN,33+IMT,ILAYER) = AL2
                  TENSTF(INTGPN,34+IMT,ILAYER) = AM2
                  TENSTF(INTGPN,35+IMT,ILAYER) = AN2
                  TENSTF(INTGPN,36+IMT,ILAYER) = AL3
                  TENSTF(INTGPN,37+IMT,ILAYER) = AM3
                  TENSTF(INTGPN,38+IMT,ILAYER) = AN3
                  ELSE
                  EPSY = EPSY1
                  ILY = ILY1
                  SIGYLD = SIGYD1
                  ARATIO = ARATO1
                  AL1 = ELLYFO(ICNT3,ELNUM)
                  AM1 = ELLYFO(ICNT3+1,ELNUM)
                  AN1 = ELLYFO(ICNT3+2,ELNUM)
                  AL2 = ELLYFO(ICNT3+3,ELNUM)
                  AM2 = ELLYFO(ICNT3+4,ELNUM)
                  AN2 = ELLYFO(ICNT3+5,ELNUM)
                  AL3 = ELLYFO(ICNT3+6,ELNUM)
                  AM3 = ELLYFO(ICNT3+7,ELNUM)
                  AN3 = ELLYFO(ICNT3+8,ELNUM)
                  ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,
     1                ILAYER) + 1
                  IPNT = ICOUNT(INTGPN,ILAYER)
                  TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
                  TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
                  TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
                  TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
                  TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
                  IMT = 9*(IPNT-1) + 1
                  TENSTF(INTGPN,30+IMT,ILAYER) = AL1
                  TENSTF(INTGPN,31+IMT,ILAYER) = AM1
                  TENSTF(INTGPN,32+IMT,ILAYER) = AN1
                  TENSTF(INTGPN,33+IMT,ILAYER) = AL2
                  TENSTF(INTGPN,34+IMT,ILAYER) = AM2
                  TENSTF(INTGPN,35+IMT,ILAYER) = AN2
                  TENSTF(INTGPN,36+IMT,ILAYER) = AL3
                  TENSTF(INTGPN,37+IMT,ILAYER) = AM3
                  TENSTF(INTGPN,38+IMT,ILAYER) = AN3
                  ENDIF
              ELSE
                  ICNT3 = ICNT1 + 9
                  EPSY = EPSY1
                  ILY = ILY1
                  SIGYLD = SIGYD1
                  ARATIO = ARATO1
                  AL1 = ELLYFO(ICNT3,ELNUM)
                  AM1 = ELLYFO(ICNT3+1,ELNUM)
                  AN1 = ELLYFO(ICNT3+2,ELNUM)
                  AL2 = ELLYFO(ICNT3+3,ELNUM)
                  AM2 = ELLYFO(ICNT3+4,ELNUM)
                  AN2 = ELLYFO(ICNT3+5,ELNUM)
                  AL3 = ELLYFO(ICNT3+6,ELNUM)
                  AM3 = ELLYFO(ICNT3+7,ELNUM)
                  AN3 = ELLYFO(ICNT3+8,ELNUM)
```

```
                        ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,
     1                      ILAYER) + 1
                        IPNT = ICOUNT(INTGPN,ILAYER)
                        TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
                        TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
                        TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
                        TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
                        TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
                        IMT = 9*(IPNT-1) + 1
                        TENSTF(INTGPN,30+IMT,ILAYER) = AL1
                        TENSTF(INTGPN,31+IMT,ILAYER) = AM1
                        TENSTF(INTGPN,32+IMT,ILAYER) = AN1
                        TENSTF(INTGPN,33+IMT,ILAYER) = AL2
                        TENSTF(INTGPN,34+IMT,ILAYER) = AM2
                        TENSTF(INTGPN,35+IMT,ILAYER) = AN2
                        TENSTF(INTGPN,36+IMT,ILAYER) = AL3
                        TENSTF(INTGPN,37+IMT,ILAYER) = AM3
                        TENSTF(INTGPN,38+IMT,ILAYER) = AN3
                    ENDIF
                ELSE
                    ICNT3 = ICNT1 + 9
                    EPSY = EPSY1
                    ILY = ILY1
                    SIGYLD = SIGYD1
                    ARATIO = ARATO1
                    AL1 = ELLYFO(ICNT3,ELNUM)
                    AM1 = ELLYFO(ICNT3+1,ELNUM)
                    AN1 = ELLYFO(ICNT3+2,ELNUM)
                    AL2 = ELLYFO(ICNT3+3,ELNUM)
                    AM2 = ELLYFO(ICNT3+4,ELNUM)
                    AN2 = ELLYFO(ICNT3+5,ELNUM)
                    AL3 = ELLYFO(ICNT3+6,ELNUM)
                    AM3 = ELLYFO(ICNT3+7,ELNUM)
                    AN3 = ELLYFO(ICNT3+8,ELNUM)
                    ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,
     1                  ILAYER) + 1
                    IPNT = ICOUNT(INTGPN,ILAYER)
                    TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
                    TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
                    TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
                    TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
                    TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
                    IMT = 9*(IPNT-1) + 1
                    TENSTF(INTGPN,30+IMT,ILAYER) = AL1
                    TENSTF(INTGPN,31+IMT,ILAYER) = AM1
                    TENSTF(INTGPN,32+IMT,ILAYER) = AN1
                    TENSTF(INTGPN,33+IMT,ILAYER) = AL2
                    TENSTF(INTGPN,34+IMT,ILAYER) = AM2
                    TENSTF(INTGPN,35+IMT,ILAYER) = AN2
                    TENSTF(INTGPN,36+IMT,ILAYER) = AL3
                    TENSTF(INTGPN,37+IMT,ILAYER) = AM3
                    TENSTF(INTGPN,38+IMT,ILAYER) = AN3
                ENDIF
            ELSE
                ICNT3 = ICNT1 + 9
                EPSY = EPSY1
                ILY = ILY1
                SIGYLD = SIGYD1
                ARATIO = ARATO1
                AL1 = ELLYFO(ICNT3,ELNUM)
                AM1 = ELLYFO(ICNT3+1,ELNUM)
                AN1 = ELLYFO(ICNT3+2,ELNUM)
                AL2 = ELLYFO(ICNT3+3,ELNUM)
```

```
                          AM2 = ELLYFO(ICNT3+4,ELNUM)
                          AN2 = ELLYFO(ICNT3+5,ELNUM)
                          AL3 = ELLYFO(ICNT3+6,ELNUM)
                          AM3 = ELLYFO(ICNT3+7,ELNUM)
                          AN3 = ELLYFO(ICNT3+8,ELNUM)
                          ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,ILAYER)
     1                         + 1
                          IPNT = ICOUNT(INTGPN,ILAYER)
                          TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
                          TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
                          TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
                          TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
                          TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
                          IMT = 9*(IPNT-1) + 1
                          TENSTF(INTGPN,30+IMT,ILAYER) = AL1
                          TENSTF(INTGPN,31+IMT,ILAYER) = AM1
                          TENSTF(INTGPN,32+IMT,ILAYER) = AN1
                          TENSTF(INTGPN,33+IMT,ILAYER) = AL2
                          TENSTF(INTGPN,34+IMT,ILAYER) = AM2
                          TENSTF(INTGPN,35+IMT,ILAYER) = AN2
                          TENSTF(INTGPN,36+IMT,ILAYER) = AL3
                          TENSTF(INTGPN,37+IMT,ILAYER) = AM3
                          TENSTF(INTGPN,38+IMT,ILAYER) = AN3
                      ENDIF
                  ELSE
                      ICNT3 = ICNT1 + 9
                      EPSY = EPSY1
                      ILY = ILY1
                      SIGYLD = SIGYD1
                      ARATIO = ARATO1
                      AL1 = ELLYFO(ICNT3,ELNUM)
                      AM1 = ELLYFO(ICNT3+1,ELNUM)
                      AN1 = ELLYFO(ICNT3+2,ELNUM)
                      AL2 = ELLYFO(ICNT3+3,ELNUM)
                      AM2 = ELLYFO(ICNT3+4,ELNUM)
                      AN2 = ELLYFO(ICNT3+5,ELNUM)
                      AL3 = ELLYFO(ICNT3+6,ELNUM)
                      AM3 = ELLYFO(ICNT3+7,ELNUM)
                      AN3 = ELLYFO(ICNT3+8,ELNUM)
                      ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,ILAYER) + 1
                      IPNT = ICOUNT(INTGPN,ILAYER)
                      TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
                      TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
                      TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
                      TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
                      TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
                      IMT = 9*(IPNT-1) + 1
                      TENSTF(INTGPN,30+IMT,ILAYER) = AL1
                      TENSTF(INTGPN,31+IMT,ILAYER) = AM1
                      TENSTF(INTGPN,32+IMT,ILAYER) = AN1
                      TENSTF(INTGPN,33+IMT,ILAYER) = AL2
                      TENSTF(INTGPN,34+IMT,ILAYER) = AM2
                      TENSTF(INTGPN,35+IMT,ILAYER) = AN2
                      TENSTF(INTGPN,36+IMT,ILAYER) = AL3
                      TENSTF(INTGPN,37+IMT,ILAYER) = AM3
                      TENSTF(INTGPN,38+IMT,ILAYER) = AN3
                  ENDIF
              ENDIF
          ENDIF
      ENDIF
C
C
C
```

```
      IF (ICOUNT(INTGPN,ILAYER) .LE. 2) THEN
          IF (ILAYER - 2.GE.1 .AND. ILAYER.LE.NLAYRS) THEN
              ICON = (ILAYER-1)*21 + 1
              MAC = ELLYFO(ICON,ELNUM)
              ES = PROPER(1,MAC)
              ICNT1 = (ILAYER-3)*21 + 1
              MATRL1 = ELLYFO(ICNT1,ELNUM)


              ICK1 = PROPER(25,MATRL1)
              IF (ICK1 .EQ. 0) THEN
                  ILY1 = ILAYER - 2
                  EPSY1 = PROPER(5,MATRL1)/PROPER(1,MATRL1)
                  SIGYD1 = PROPER(5,MATRL1)
                  ARATO1 = PROPER(1,MATRL1)/ES
                  ICNT2 = (ILAYER-2)*21 + 1
                  MATRL2 = ELLYFO(ICNT2,ELNUM)
                  ICK2 = PROPER(25,MATRL2)
                  IF (ICK2 .EQ. 0) THEN
                      IF (ILAYER + 1 .LE. NLAYRS) THEN
                          ICNT3 = ILAYER*21 + 1
                          MATRL3 = ELLYFO(ICNT3,ELNUM)
                          ICK3 = PROPER(25,MATRL3)
                          IF (ICK3 .EQ. 0) THEN
                              ILY3 = ILAYER + 1
                              EPSY3 = PROPER(5,MATRL3)/PROPER(1,MATRL3)
                              SIGYD3 = PROPER(5,MATRL3)
                              ARATO3 = PROPER(1,MATRL3)/ES
                              ICNT4 = ICNT1 + 9
                              ICNT5 = ICNT3 + 9
                              AL1 = ELLYFO(ICNT4,ELNUM)
                              AM1 = ELLYFO(ICNT4+1,ELNUM)
                              AN1 = ELLYFO(ICNT4+2,ELNUM)
                              AL2 = ELLYFO(ICNT5,ELNUM)
                              AM2 = ELLYFO(ICNT5+1,ELNUM)
                              AN2 = ELLYFO(ICNT5+2,ELNUM)
                              DOT = AL1*AL2 + AM1*AM2 + AN1*AN2
                              IF (DABS(DOT).GE.0.999 .AND. DABS(DOT).LE.
     1                            1.001) THEN
                                  IF (EPSY3 .GT. EPSY1) THEN
                                  EPSY = EPSY3
                                  ILY = ILY3
                                  SIGYLD = SIGYD3
                                  ARATIO = ARATO3
                                  AL1 = ELLYFO(ICNT5,ELNUM)
                                  AM1 = ELLYFO(ICNT5+1,ELNUM)
                                  AN1 = ELLYFO(ICNT5+2,ELNUM)
                                  AL2 = ELLYFO(ICNT5+3,ELNUM)
                                  AM2 = ELLYFO(ICNT5+4,ELNUM)
                                  AN2 = ELLYFO(ICNT5+5,ELNUM)
                                  AL3 = ELLYFO(ICNT5+6,ELNUM)
                                  AM3 = ELLYFO(ICNT5+7,ELNUM)
                                  AN3 = ELLYFO(ICNT5+8,ELNUM)
                                  ELSE
                                  EPSY = EPSY1
                                  ILY = ILY1
                                  SIGYLD = SIGYD1
                                  ARATIO = ARATO1
                                  AL1 = ELLYFO(ICNT4,ELNUM)
                                  AM1 = ELLYFO(ICNT4+1,ELNUM)
                                  AN1 = ELLYFO(ICNT4+2,ELNUM)
                                  AL2 = ELLYFO(ICNT4+3,ELNUM)
                                  AM2 = ELLYFO(ICNT4+4,ELNUM)
                                  AN2 = ELLYFO(ICNT4+5,ELNUM)
```

```
                  AL3 = ELLYFO(ICNT4+6,ELNUM)
                  AM3 = ELLYFO(ICNT4+7,ELNUM)
                  AN3 = ELLYFO(ICNT4+8,ELNUM)
                  ENDIF
                  ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,
     1                ILAYER) + 1
                  IPNT = ICOUNT(INTGPN,ILAYER)
                  TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
                  TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
                  TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
                  TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
                  TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
                  IMT = 9*(IPNT-1) + 1
                  TENSTF(INTGPN,30+IMT,ILAYER) = AL1
                  TENSTF(INTGPN,31+IMT,ILAYER) = AM1
                  TENSTF(INTGPN,32+IMT,ILAYER) = AN1
                  TENSTF(INTGPN,33+IMT,ILAYER) = AL2
                  TENSTF(INTGPN,34+IMT,ILAYER) = AM2
                  TENSTF(INTGPN,35+IMT,ILAYER) = AN2
                  TENSTF(INTGPN,36+IMT,ILAYER) = AL3
                  TENSTF(INTGPN,37+IMT,ILAYER) = AM3
                  TENSTF(INTGPN,38+IMT,ILAYER) = AN3
              ELSE IF (ILAYER + 2 .LE. NLAYRS) THEN
                  ICNT6 = (ILAYER+1)*21 + 1
                  MATRL4 = ELLYFO(ICNT6,ELNUM)
                  ICK4 = PROPER(25,MATRL4)
                  IF (ICK4 .EQ. 0) THEN
                  ILY4 = ILAYER + 2
                  EPSY4 = PROPER(5,MATRL4)/PROPER(1,
     1                MATRL4)
                  ARATO4 = PROPER(1,MATRL4)/ES
                  SIGYD4 = PROPER(5,MATRL4)
                  ICNT3 = ICNT1 + 9
                  ICNT6 = ICNT5 + 9
                  AL1 = ELLYFO(ICNT3,ELNUM)
                  AM1 = ELLYFO(ICNT3+1,ELNUM)
                  AN1 = ELLYFO(ICNT3+2,ELNUM)
                  AL2 = ELLYFO(ICNT6,ELNUM)
                  AM2 = ELLYFO(ICNT6+1,ELNUM)
                  AN2 = ELLYFO(ICNT6+2,ELNUM)
                  DOT = AL1*AL2 + AM1*AM2 + AN1*AN2
                  IF (DABS(DOT).GE.0.999 .AND. DABS(DOT)
     1                .LE.1.001) THEN
                  IF (EPSY4 .GT. EPSY1) THEN
                  EPSY = EPSY4
                  ILY = ILY4
                  SIGYLD = SIGYD4
                  ARATIO = ARATO4
                  AL1 = ELLYFO(ICNT6,ELNUM)
                  AM1 = ELLYFO(ICNT6+1,ELNUM)
                  AN1 = ELLYFO(ICNT6+2,ELNUM)
                  AL2 = ELLYFO(ICNT6+3,ELNUM)
                  AM2 = ELLYFO(ICNT6+4,ELNUM)
                  AN2 = ELLYFO(ICNT6+5,ELNUM)
                  AL3 = ELLYFO(ICNT6+6,ELNUM)
                  AM3 = ELLYFO(ICNT6+7,ELNUM)
                  AN3 = ELLYFO(ICNT6+8,ELNUM)
                  ELSE
                  EPSY = EPSY1
                  ILY = ILY1
                  ARATIO = ARATO1
                  SIGYLD = SIGYD1
                  AL1 = ELLYFO(ICNT4,ELNUM)
```

```
      AM1 = ELLYFO(ICNT4+1,ELNUM)
      AN1 = ELLYFO(ICNT4+2,ELNUM)
      AL2 = ELLYFO(ICNT4+3,ELNUM)
      AM2 = ELLYFO(ICNT4+4,ELNUM)
      AN2 = ELLYFO(ICNT4+5,ELNUM)
      AL3 = ELLYFO(ICNT4+6,ELNUM)
      AM3 = ELLYFO(ICNT4+7,ELNUM)
      AN3 = ELLYFO(ICNT4+8,ELNUM)
      ENDIF
      ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,
     1    ILAYER) + 1
      IPNT = ICOUNT(INTGPN,ILAYER)
      TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
      TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
      TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
      TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
      TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
      IMT = 9*(IPNT-1) + 1
      TENSTF(INTGPN,30+IMT,ILAYER) = AL1
      TENSTF(INTGPN,31+IMT,ILAYER) = AM1
      TENSTF(INTGPN,32+IMT,ILAYER) = AN1
      TENSTF(INTGPN,33+IMT,ILAYER) = AL2
      TENSTF(INTGPN,34+IMT,ILAYER) = AM2
      TENSTF(INTGPN,35+IMT,ILAYER) = AN2
      TENSTF(INTGPN,36+IMT,ILAYER) = AL3
      TENSTF(INTGPN,37+IMT,ILAYER) = AM3
      TENSTF(INTGPN,38+IMT,ILAYER) = AN3
      ELSE
      ICNT3 = ICNT1 + 9
      EPSY = EPSY1
      ILY = ILY1
      SIGYLD = SIGYD1
      ARATIO = ARATO1
      AL1 = ELLYFO(ICNT3,ELNUM)
      AM1 = ELLYFO(ICNT3+1,ELNUM)
      AN1 = ELLYFO(ICNT3+2,ELNUM)
      AL2 = ELLYFO(ICNT3+3,ELNUM)
      AM2 = ELLYFO(ICNT3+4,ELNUM)
      AN2 = ELLYFO(ICNT3+5,ELNUM)
      AL3 = ELLYFO(ICNT3+6,ELNUM)
      AM3 = ELLYFO(ICNT3+7,ELNUM)
      AN3 = ELLYFO(ICNT3+8,ELNUM)
      ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,
     1    ILAYER) + 1
      IPNT = ICOUNT(INTGPN,ILAYER)
      TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
      TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
      TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
      TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
      TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
      IMT = 9*(IPNT-1) + 1
      TENSTF(INTGPN,30+IMT,ILAYER) = AL1
      TENSTF(INTGPN,31+IMT,ILAYER) = AM1
      TENSTF(INTGPN,32+IMT,ILAYER) = AN1
      TENSTF(INTGPN,33+IMT,ILAYER) = AL2
      TENSTF(INTGPN,34+IMT,ILAYER) = AM2
      TENSTF(INTGPN,35+IMT,ILAYER) = AN2
      TENSTF(INTGPN,36+IMT,ILAYER) = AL3
      TENSTF(INTGPN,37+IMT,ILAYER) = AM3
      TENSTF(INTGPN,38+IMT,ILAYER) = AN3
      ENDIF
      ELSE
      EPSY = EPSY1
```

```
                  ILY = ILY1
                  SIGYLD = SIGYD1
                  ARATIO = ARATO1
                  ICNT3 = ICNT1 + 9
                  AL1 = ELLYFO(ICNT3,ELNUM)
                  AM1 = ELLYFO(ICNT3+1,ELNUM)
                  AN1 = ELLYFO(ICNT3+2,ELNUM)
                  AL2 = ELLYFO(ICNT3+3,ELNUM)
                  AM2 = ELLYFO(ICNT3+4,ELNUM)
                  AN2 = ELLYFO(ICNT3+5,ELNUM)
                  AL3 = ELLYFO(ICNT3+6,ELNUM)
                  AM3 = ELLYFO(ICNT3+7,ELNUM)
                  AN3 = ELLYFO(ICNT3+8,ELNUM)
                  ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,
     1                ILAYER) + 1

                  IPNT = ICOUNT(INTGPN,ILAYER)
                  TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
                  TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
                  TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
                  TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
                  TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
                  IMT = 9*(IPNT-1) + 1
                  TENSTF(INTGPN,30+IMT,ILAYER) = AL1
                  TENSTF(INTGPN,31+IMT,ILAYER) = AM1
                  TENSTF(INTGPN,32+IMT,ILAYER) = AN1
                  TENSTF(INTGPN,33+IMT,ILAYER) = AL2
                  TENSTF(INTGPN,34+IMT,ILAYER) = AM2
                  TENSTF(INTGPN,35+IMT,ILAYER) = AN2
                  TENSTF(INTGPN,36+IMT,ILAYER) = AL3
                  TENSTF(INTGPN,37+IMT,ILAYER) = AM3
                  TENSTF(INTGPN,38+IMT,ILAYER) = AN3
                  ENDIF
              ELSE
                  EPSY = EPSY1
                  ILY = ILY1
                  SIGYLD = SIGYD1
                  ARATIO = ARATO1
                  ICNT3 = ICNT1 + 9
                  AL1 = ELLYFO(ICNT3,ELNUM)
                  AM1 = ELLYFO(ICNT3+1,ELNUM)
                  AN1 = ELLYFO(ICNT3+2,ELNUM)
                  AL2 = ELLYFO(ICNT3+3,ELNUM)
                  AM2 = ELLYFO(ICNT3+4,ELNUM)
                  AN2 = ELLYFO(ICNT3+5,ELNUM)
                  AL3 = ELLYFO(ICNT3+6,ELNUM)
                  AM3 = ELLYFO(ICNT3+7,ELNUM)
                  AN3 = ELLYFO(ICNT3+8,ELNUM)
                  ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,
     1                ILAYER) + 1
                  IPNT = ICOUNT(INTGPN,ILAYER)
                  TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
                  TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
                  TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
                  TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
                  TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
                  IMT = 9*(IPNT-1) + 1
                  TENSTF(INTGPN,30+IMT,ILAYER) = AL1
                  TENSTF(INTGPN,31+IMT,ILAYER) = AM1
                  TENSTF(INTGPN,32+IMT,ILAYER) = AN1
                  TENSTF(INTGPN,33+IMT,ILAYER) = AL2
                  TENSTF(INTGPN,34+IMT,ILAYER) = AM2
                  TENSTF(INTGPN,35+IMT,ILAYER) = AN2
```

```
                    TENSTF(INTGPN,36+IMT,ILAYER) = AL3
                    TENSTF(INTGPN,37+IMT,ILAYER) = AM3
                    TENSTF(INTGPN,38+IMT,ILAYER) = AN3
                ENDIF
            ELSE
                EPSY = EPSY1
                ILY = ILY1
                SIGYLD = SIGYD1
                ARATIO = ARATO1
                ICNT3 = ICNT1 + 9
                AL1 = ELLYFO(ICNT3,ELNUM)
                AM1 = ELLYFO(ICNT3+1,ELNUM)
                AN1 = ELLYFO(ICNT3+2,ELNUM)
                AL2 = ELLYFO(ICNT3+3,ELNUM)
                AM2 = ELLYFO(ICNT3+4,ELNUM)
                AN2 = ELLYFO(ICNT3+5,ELNUM)
                AL3 = ELLYFO(ICNT3+6,ELNUM)
                AM3 = ELLYFO(ICNT3+7,ELNUM)
                AN3 = ELLYFO(ICNT3+8,ELNUM)
                ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,
     1              ILAYER) + 1
                IPNT = ICOUNT(INTGPN,ILAYER)
                TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
                TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
                TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
                TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
                TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
                IMT = 9*(IPNT-1) + 1
                TENSTF(INTGPN,30+IMT,ILAYER) = AL1
                TENSTF(INTGPN,31+IMT,ILAYER) = AM1
                TENSTF(INTGPN,32+IMT,ILAYER) = AN1
                TENSTF(INTGPN,33+IMT,ILAYER) = AL2
                TENSTF(INTGPN,34+IMT,ILAYER) = AM2
                TENSTF(INTGPN,35+IMT,ILAYER) = AN2
                TENSTF(INTGPN,36+IMT,ILAYER) = AL3
                TENSTF(INTGPN,37+IMT,ILAYER) = AM3
                TENSTF(INTGPN,38+IMT,ILAYER) = AN3
            ENDIF
        ELSE
            EPSY = EPSY1
            ILY = ILY1
            SIGYLD = SIGYD1
            ARATIO = ARATO1
            ICNT3 = ICNT1 + 9
            AL1 = ELLYFO(ICNT3,ELNUM)
            AM1 = ELLYFO(ICNT3+1,ELNUM)
            AN1 = ELLYFO(ICNT3+2,ELNUM)
            AL2 = ELLYFO(ICNT3+3,ELNUM)
            AM2 = ELLYFO(ICNT3+4,ELNUM)
            AN2 = ELLYFO(ICNT3+5,ELNUM)
            AL3 = ELLYFO(ICNT3+6,ELNUM)
            AM3 = ELLYFO(ICNT3+7,ELNUM)
            AN3 = ELLYFO(ICNT3+8,ELNUM)
            ICOUNT(INTGPN,ILAYER) = ICOUNT(INTGPN,ILAYER)
     1           + 1
            IPNT = ICOUNT(INTGPN,ILAYER)
            TENSTF(INTGPN,3+IPNT,ILAYER) = IPNT
            TENSTF(INTGPN,12+IPNT,ILAYER) = EPSY
            TENSTF(INTGPN,66+IPNT,ILAYER) = ILY
            TENSTF(INTGPN,60+IPNT,ILAYER) = ARATIO
            TENSTF(INTGPN,63+IPNT,ILAYER) = SIGYLD
            IMT = 9*(IPNT-1) + 1
            TENSTF(INTGPN,30+IMT,ILAYER) = AL1
```

```
                        TENSTF(INTGPN,31+IMT,ILAYER) = AM1
                        TENSTF(INTGPN,32+IMT,ILAYER) = AN1
                        TENSTF(INTGPN,33+IMT,ILAYER) = AL2
                        TENSTF(INTGPN,34+IMT,ILAYER) = AM2
                        TENSTF(INTGPN,35+IMT,ILAYER) = AN2
                        TENSTF(INTGPN,36+IMT,ILAYER) = AL3
                        TENSTF(INTGPN,37+IMT,ILAYER) = AM3
                        TENSTF(INTGPN,38+IMT,ILAYER) = AN3
                     ENDIF
                  ENDIF
               ENDIF
            ENDIF
         ENDIF
         RETURN
         END

C ---------------------------  S  U  B  C  O  N  ----------------------
C
C     INCLUDE(PROCESS)
      SUBROUTINE SUBCON(PDIR,SIGMA1,SIGMA2,SIGMA3,PROPER,L,J,CONVER,
     $                  IFLAG1,IFLAG2,IFLAG3,EPS1,EPS2,EPS3,IFC)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C STATUS
C
C************************************************************
      COMMON /BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON /ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      DIMENSION PROPER(25),PDIR(3,3)
C
C
C              SUBROUTINE TO CHECK THE GAUSS POINT 'L' OF CONCRETE
C              LAYER 'J' FOR THE CURRENT ELEMENT. THIS CHECK IS DONE
C              FOR A POINT WHICH WAS PREVIOUSLY UNLOADING OR RELOADING
C              AND IN THE CURRENT ITERATION IS DETERMINED AS BEING IN
C              LOADING CONDITION (INCREASE IN STRAIN OF AT LEAST ONE
C              DIRECTION) OR FOR A POINT SHOWING STRAINS WITH DIFFERNT
C              SIGNS IN THE CURRENT AND PREVIOUS ITERATION.
C
C
      HISTO(L,10,J) = PDIR(1,1)
      HISTO(L,11,J) = PDIR(1,2)
      HISTO(L,12,J) = PDIR(1,3)
      HISTO(L,13,J) = PDIR(2,1)
      HISTO(L,14,J) = PDIR(2,2)
      HISTO(L,15,J) = PDIR(2,3)
      HISTO(L,16,J) = PDIR(3,1)
      HISTO(L,17,J) = PDIR(3,2)
      HISTO(L,18,J) = PDIR(3,3)
C
C              DETERMINE  STATE OF STRESS AND RATIO OF PRINC. STRESSES
C
      CALL STATUS (SIGMA1, SIGMA2, SIGMA3, STATE, J, PROPER, L, IFC)
      HISTO(L,25,J) = STATE
  100 CONTINUE
      IF (IFLAG1.EQ.1 .OR. IFLAG2.EQ.1 .OR. IFLAG3.EQ.1) THEN
C
```

```
C               STRAINING IN THE OPPOSITE DIRECTION ....
C
          IF (IFLAG1 .EQ. 1) THEN
C
C               LET THE CURRENT STRAIN REBOUND UP TO ZERO VALUE
C               FOR THE TIME BEING.
C
              HISTO(L,31,J) = 0.0
              HISTO(L,22,J) = 0.0
          ENDIF
C
          IF (IFLAG2 .EQ. 1) THEN
              HISTO(L,32,J) = 0.0
              HISTO(L,23,J) = 0.0
          ENDIF
          IF (IFLAG3 .EQ. 1) THEN
              HISTO(L,33,J) = 0.0
              HISTO(L,24,J) = 0.0
          ENDIF
C
C               SET THE STATE OF THE POINT TO LOADING AND CARRY OUT
C               FURTHER ITERATIONS.
C
          HISTO(L,30,J) = 1.
          CONVER = 999.0
          RETURN
      ENDIF
C
C
C               THE POINT IS RELOADING.ITS SECANT 'E' WILL REMAIN
C               UNCHANGED UP TO THE STRAIN FROM WHICH UNLOADING
C               INITIATED.FROM THERE ON THE VIRGIN STRESS-STRAIN
C               CURVE WILL BE FOLLOWED.
C
C
      IF (DABS(EPS1) .GE. .999*DABS(HISTO(L,31,J))) THEN
C
C
C               LET THE CURRENT STRAIN BE EQUAL TO THE STRAIN AT THE
C               INITIATION OF UNLOADING AND CARRY OUT FURTHER ITERATION
C
C
          HISTO(L,22,J) = HISTO(L,31,J)
C
C               NOW THE POINT IS IN THE STATE OF LOADING AGAIN.
C
          HISTO(L,30,J) = 1.
          CONVER = 999.0
      ENDIF
      IF (DABS(EPS2) .GE. .999*DABS(HISTO(L,32,J))) THEN
          HISTO(L,23,J) = HISTO(L,32,J)
          HISTO(L,30,J) = 1.
          CONVER = 999.
      ENDIF
      IF (DABS(EPS3) .GE. .999*DABS(HISTO(L,33,J))) THEN
          HISTO(L,24,J) = HISTO(L,33,J)
          HISTO(L,30,J) = 1.
          CONVER = 999.
      ENDIF
      RETURN
      END
C
C
```

```
C
C
C ------------------------- C O N 1 2 ---------------------------
C
C     INCLUDE(PROCESS)
      SUBROUTINE CON12(SIGMA1,SIGMA2,SIGMA3,L,J,EPS1,EPS2,EPS3,NINC,
     $ NIT,PROPER,ELNUM,NNEL,IOUT,CONVER,NSTIFF,TOLER,PDIR,CHKGUS,ITYPE
     $ ,DTSTIF,IFC,IERROR,IFLG1,IFLG2,IFLG3,ITCRK,ICOUP,IFLAG8)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C STATUS    FAILUR    FINDEX    MATERL    PRECON    CKINIT
C
C*********************************************************
      INTEGER ELNUM
      COMMON /BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON /ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      COMMON/TRANS/V1(3),V2(3),V3(3)
      DIMENSION PROPER(25)
      REAL*8 NUS,PDIR(3,3),DTSTIF(6,6)
      LOGICAL CHKGUS
C
C
C      SUBROUTINE TO CHECK CONCRETE MATERIAL LAW IN THE
C      PRE-ULTIMATE PHASE.
C
C      PREVENT OVERSHOOTING WHEN THE STRESS POINT IS IN THE
C      PRE-ULTIMATE STAGE AND LOADING.
C
C      ..... TOLERE = VERY SMALL INTERNALLY SET TOLERANCE FOR
C                     MODULUS OF ELSTICITY WHEN OVERSHOOTING
C                     IS TAKING PLACE.
C      ..... TOLERN = VERY SMALL, INTERNALLY SET, TOLERANCE
C                     FOR POISSON'S RATIO WHEN THERE IS
C                     OVERSHOOTING.
C
C
      TOLERE = TOLER
      TOLERN = TOLER
C
C
C      OVRSHT = 0.0 .... THE STRESS POINT HAS NOT REACHED ITS
C                        ULTIMATE VALUE WITH THE CURRENT RATIO
C
C      OVRSHT = 1.0 .... THE STRESS POINT HAS JUST REACHED OR
C                                  EXCEEDED ITS ULTIMATE VALUE.
C
C     INITIALIZE FUNCT,OVRSHT
C
      C1 = 0.0
      C2 = 0.0
      C3 = 0.0
      FUNCT = 0.0
      STATE = 0.0
      OVRSHT = 0.0
      ES = 0.0
      NUS = 0.0
C
```

```
C              CALCULATE THE RATIO OF MAX. TO MIN. PRINCIPAL
C              STRESS AND THE STATE OF STRESS.
C
C
      PI = 3.141592654
      FC = PROPER(5)
      FT = PROPER(6)
      CALL STATUS (SIGMA1, SIGMA2, SIGMA3, STATE, J, PROPER, L, IFC)
      IF (CRACK(L,1,J) .GE. 1.0) THEN
          IF (IFC .EQ. 2) THEN
              IF (HISTO(L,41,J) .LT. PROPER(7)) THEN
                  FC = FC*(1.0-0.2*(HISTO(L,41,J)/PROPER(7)))
              ELSE
                  FC = FC*0.8
              ENDIF
              FC = DMIN1(HISTO(L,58,J),FC)
              HISTO(L,58,J) = FC
          ENDIF
      ENDIF
C
C               SAVE CURRENT STATE OF TRIAXIAL STRESS
C
      HISTO(L,25,J) = STATE
      IF (STATE.EQ.100. .OR. STATE.EQ.110. .OR. STATE.EQ.111) THEN
C
C              STATE OF UNIAXIAL OR BI/TRI AXIAL TENSION.
C
          IF (SIGMA1 .GE. FT) THEN
C
C
C              THE STRESS POINT HAS JUST REACHED OR VIOLATED THE
C              FAILURE ENVELOPE.TO AVOID OVERSHOOTING FORCE THE
C              POINT TO BE ON THE FAILURE ENVELOPE.
C
C
              OVRSHT = 1.0
              HISTO(L,29,J) = 2.
C
C
C              NOW CALCULATE SECANT VALUES AT FAILURE USING THE RATIO
C              OF PRINCIPAL STRESSES DETERMINED AT THE PREVIOUS ITER-
C              ATION.
C
C
              CALL FAILUR (L, J, PROPER, IOUT, ES, NUS, SIGMA1, SIGMA2,
     1            SIGMA3, IFC, ELNUM)
              GO TO 160
          ENDIF
          GO TO 150
      ENDIF
C
C              STATE OF TENSION-COMPRESSION (CRACKING ZONE)
C
      IF (STATE .EQ. 101.) CALL FINDEX (FUNCT, FC, PROPER, SIGMA1,
     1    SIGMA2, SIGMA3, IOUT)
C
C              STATE OF TENSION-COMPRESSION (CRUSHING ZONE)
C
      IF (STATE .EQ. (-101.)) CALL FINDEX (FUNCT, FC, PROPER, SIGMA1,
     1    SIGMA2, SIGMA3, IOUT)
C
C        STATE OF UNIAXIAL, BIAXIAL OR TRIAXIAL COMPRESSION
C
```

```
      IF (STATE.EQ.(-100.) .OR. STATE.EQ.(-110.) .OR. STATE.EQ.(-111.))
     1    CALL FINDEX (FUNCT, FC, PROPER, SIGMA1, SIGMA2, SIGMA3, IOUT)
      IF (STATE.NE.100. .AND. STATE.NE.110. .AND. STATE.NE.111.) HISTO(L
     1    ,54,J) = FUNCT
      IF (FUNCT .GE. (-.01)) THEN
C
C
C      THE STRESS POINT HAS JUST REACHED OR VIOLATED THE
C      FAILURE ENVELOPE.FORCE THE STRESS POINT TO BE ON
C      THE FAILURE SURFACE FOR THE CURRENT ITERATION.
C
C
          OVRSHT = 1.0
          HISTO(L,29,J) = 2.
C
C                 CALCULATE THE SECANT VALUES AT FAILURE FOR THE POINT.
C
          CALL FAILUR (L, J, PROPER, IOUT, ES, NUS, SIGMA1, SIGMA2,
     1        SIGMA3, IFC, ELNUM)
C
C
C       THERE IS A POSSIBILITY THAT OVERSHOOTING AROUND THE
C             ULTIAMATE POINT IS TAKING PLACE.COMPARE THE CURRENT
C             PRINCIPAL STRESSES WITH THE CALCULATED VALUES AT
C             ULTIMATE.IF THE DIFFERENCE IS WITHIN 2% THEN USE A
C             VERY SMALL TOLERANCE OF 2.0% FOR E TO AVOID EXCESSIVE
C             SOFTENING THAT WILL CAUSE CONVERGENCE PROBLEMS.THE
C             SELECTED TOLERANCE WILL ENSURE THAT CONVERGENCE IS
C             ACHIEVED FROM THE STIFF SIDE.THIS TOLERANCE WILL BE
C             IN EFFECT FOR THE CURRENT LOAD INCREMENT.
C


C             ALSO SET THE TOLERANCE FOR POISSON'S RATIO TO 2.0%
C             THIS TOLERANCE EFFECTS THE ADJUSTED VALUE DETERMINED
C             FOR THE MODULUS OF ELASTICITY IN SUB. 'PRECON'.
C
C
          C1 = 3.
          C2 = 3.
          C3 = 3.
          SIGM1F = HISTO(L,34,J)
          SIGM2F = HISTO(L,35,J)
          SIGM3F = HISTO(L,36,J)
          IF (STATE .EQ. 100) GO TO 130
          IF (STATE .EQ. 110) GO TO 110
          IF (DABS(SIGM3F) .EQ. 0.0) GO TO 100
          C3 = 100.*(DABS(SIGMA3)-DABS(SIGM3F))/DABS(SIGM3F)
  100     CONTINUE
          IF (STATE .EQ. (-100)) GO TO 140
  110     CONTINUE
          IF (DABS(SIGM2F) .EQ. 0.0) GO TO 120
          C2 = 100.*(DABS(SIGMA2)-DABS(SIGM2F))/DABS(SIGM2F)
  120     CONTINUE
          IF (STATE .EQ. (-110)) GO TO 140
  130     CONTINUE
          IF (DABS(SIGM1F) .EQ. 0.0) GO TO 140
          C1 = 100.*(DABS(SIGMA1)-DABS(SIGM1F))/DABS(SIGM1F)
  140     CONTINUE
          IF (C1.LE.2. .OR. C2.LE.2. .OR. C3.LE.2.) THEN
              TOLERE = 2.0
              TOLERN = 2.0
          ENDIF
          GO TO 160
```

```
      ENDIF
C
C               CALL THE MATERIAL SUBROUTINE TO DETERMINE CURRENT
C               VALUES OF SECANT 'E' AND 'NU'.
C
C
  150 CONTINUE
      CALL MATERL (L, J, SIGMA1, SIGMA2, SIGMA3, STATE, ES, NUS, IOUT,
     1    PROPER, BETHA, IFC, ELNUM)
  160 CONTINUE
      HISTO(L,10,J) = PDIR(1,1)
      HISTO(L,11,J) = PDIR(1,2)
      HISTO(L,12,J) = PDIR(1,3)
      HISTO(L,13,J) = PDIR(2,1)
      HISTO(L,14,J) = PDIR(2,2)
      HISTO(L,15,J) = PDIR(2,3)
      HISTO(L,16,J) = PDIR(3,1)
      HISTO(L,17,J) = PDIR(3,2)
      HISTO(L,18,J) = PDIR(3,3)
      IF (NINC .EQ. 1) THEN
C
C
C               FOR THE FIRST LOAD STEP THE STRESS POINT MUST BE
C               INSIDE THE FAILURE ENVELOPE.
C
C
          IF (STATE.EQ.100. .OR. STATE.EQ.110. .OR. STATE.EQ.111.) THEN
              IF (SIGMA1 .GT. FT) THEN
                  WRITE (IOUT, 900) SIGMA1, FT
                  WRITE (IOUT, 910) L, J, ELNUM
                  STOP
              ENDIF
          ENDIF
          IF (FUNCT .GT. 0.0) THEN
              WRITE (IOUT, 900) SIGMA1, FT
              WRITE (IOUT, 910) L, J, ELNUM
              STOP
          ENDIF
      ENDIF
C
C     CHECK CONVERGENCE FOR E AND NU
C
C
      CALL PRECON (ES, NUS, CONVER, NSTIFF, TOLERN, TOLERE, CHKGUS, L, J
     1     , IOUT, OVRSHT, ELNUM)
C
      IF (HISTO(L,29,J) .EQ. 2.) THEN
          IF (STATE.EQ.100. .OR. STATE.EQ.110 .OR. STATE.EQ.111 .OR.
     1        STATE.EQ.101) THEN
C
C                         POSSIBILITY OF A NEW CRACK
C
C       IN ORDER TO HAVE A NEW CRACK THE ANGLE BETWEEN THE
C       CURRENT PRINCIPAL DIRECTION, THETA, AND ANY EXISTING
C       CRACK MUST BE LARGER THAN A THRESHOLD VALUE SPECIFIED
C       EXTERNALLY. THIS IS DONE TO AVOID 'NUMERICAL CRACKING'.
C
C
C
C                RECOVER THE THRESHOLD ANGLE 'ALFA'
C
              ALFA = PROPER(15)*3.1415926535/180.
C      ALFA=60 * 3.1415926535/180.
```

```
              A = DSIN(ALFA)
C
              NCRACK = INT(CRACK(L,1,J))
              IF (NCRACK .GE. 1) THEN
C
C                IF THE ANGLE BETWEEN CURRENT PRINCIPAL DIRECTION AND AN
C                NORMAL TO THE EXISTING CRACKS IS LESS THAN 'ALFA' DO NO
C                INITIATE A NEW CRACK.
C
C
                 AL1 = HISTO(L,10,J)
                 AM1 = HISTO(L,11,J)
                 AN1 = HISTO(L,12,J)
                 DO 170 ICR = 1, NCRACK
                     IMP = 9*(ICR-1) + 1
                     ACL = CRACK(L,137+IMP,J)
                     ACM = CRACK(L,138+IMP,J)
                     ACN = CRACK(L,139+IMP,J)
                     DOT = AL1*ACL + AM1*ACM + AN1*ACN
                     IF (DOT .GT. 1.0) DOT = 1.0
                     IF (DOT .LT. (-1.0)) DOT = -1.0
                     B = DSQRT(1.0D00-DOT*DOT)
                     IF (B .LT. A) THEN
C
C               ... NO NEW CRACK
C
                         HISTO(L,29,J) = 1.0
                         RETURN
                     ENDIF
  170            CONTINUE
              ENDIF
              IF (IFLAG8 .EQ. 4) THEN
                  DOT1=V3(1)*PDIR(1,1)+V3(2)*PDIR(1,2)+V3(3)*PDIR(1,3)
                  IF (DABS(DOT1) .GE. 0.99) RETURN
              ENDIF
C
C
C               INITIATION OF ANOTHER CRACK.STORE THE INFORMATION
C
C
C
              CALL CKINIT (L, J, PROPER, ELNUM, NNEL, IOUT, ITYPE,
     1            DTSTIF, IERROR, IFLG1, IFLG2, IFLG3, ITCRK, ICOUP,
     2            IFLAG8)
C
              IF (STATE .EQ. 110.) THEN
C
C                GET RATIO OF MAX. TO MIN. PRINCIPAL TENSILE STRESSES
C
                  RATIO = SIGMA2/SIGMA1
C
C                FOR EQUAL BIAXIAL TENSION TWO ORTHOGONAL CRACKS CAN
C                INITIATE AT THE SAME TIME. CHANGE
C
                  IF (RATIO.GE..99999 .AND. RATIO.LT.1.00001) THEN
C
                     AL1 = PDIR(2,1)
                     AM1 = PDIR(2,2)
                     AN1 = PDIR(2,3)
                     DO 180 ICR = 1, NCRACK
                         IMP = 9*(ICR-1) + 1
                         ACL = CRACK(L,137+IMP,J)
                         ACM = CRACK(L,138+IMP,J)
```

```
                     ACN = CRACK(L,139+IMP,J)
                     DOT = AL1*ACL + AM1*ACM + AN1*ACN
                     IF (DOT .GT. 1.0) DOT = 1.0
                     IF (DOT .LT. (-1.0)) DOT = -1.0
                     B = DSQRT(1.0D00-DOT*DOT)
                     IF (B .LT. A) THEN
C
C               ...     NO NEW CRACK
C
                        HISTO(L,29,J) = 1.0
                        GO TO 190
                     ENDIF
  180             CONTINUE
                  IF (IFLAG8 .EQ. 4) THEN
                     DOT2 = V3(1)*PDIR(2,1) + V3(2)*PDIR(2,2) + V3(
     1                  3)*PDIR(2,3)
                     IF (DABS(DOT2) .GE. 0.99) GO TO 190
                  ENDIF
C
C               ROTATE INTACT CONCRETE'S PRINCIPAL AXES TO THE OTHER
C               PRINCIPAL DIRECTION AND FORM A CRACK ORTHOGONAL TO IT.
C
                  HISTO(L,10,J) = PDIR(2,1)
                  HISTO(L,11,J) = PDIR(2,2)
                  HISTO(L,12,J) = PDIR(2,3)
                  HISTO(L,13,J) = PDIR(3,1)
                  HISTO(L,14,J) = PDIR(3,2)
                  HISTO(L,15,J) = PDIR(3,3)
                  HISTO(L,16,J) = PDIR(1,1)
                  HISTO(L,17,J) = PDIR(1,2)
                  HISTO(L,18,J) = PDIR(1,3)
C
                  CALL CKINIT (L, J, PROPER, ELNUM, NNEL, IOUT,
     1               ITYPE, DTSTIF, IERROR, IFLG1, IFLG2, IFLG3,
     2               ITCRK, ICOUP, IFLAG8)
               ENDIF
            ENDIF
  190       CONTINUE
            IF (STATE .EQ. 111.) THEN
C
C               GET RATIO OF MAX. TO MIN. PRINCIPAL TENSILE STRESSES
C
                RATIO = SIGMA3/SIGMA1
C
C               FOR EQUAL TRIAXIAL TENSION THREE ORTHOGONAL CRACKS CAN
C               INITIATE AT THE SAME TIME. CHANGE
C
                IF (RATIO.GE..99999 .AND. RATIO.LT.1.00001) THEN
C
                  AL1 = PDIR(3,1)
                  AM1 = PDIR(3,2)
                  AN1 = PDIR(3,3)
                  DO 200 ICR = 1, NCRACK
                     IMP = 9*(ICR-1) + 1
                     ACL = CRACK(L,137+IMP,J)

                     ACM = CRACK(L,138+IMP,J)
                     ACN = CRACK(L,139+IMP,J)
                     DOT = AL1*ACL + AM1*ACM + AN1*ACN
                     IF (DOT .GT. 1.0) DOT = 1.0
                     IF (DOT .LT. (-1.0)) DOT = -1.0
                     B = DSQRT(1.0D00-DOT*DOT)
                     IF (B .LT. A) THEN
```

```
C
C                ...     NO NEW CRACK
C
                               HISTO(L,29,J) = 1.0
                               GO TO 210
                           ENDIF
  200                  CONTINUE
                       IF (IFLAG8 .EQ. 4) THEN
                           DOT3 = V3(1)*PDIR(3,1) + V3(2)*PDIR(3,2) + V3(
     1                         3)*PDIR(3,3)
                           IF (DABS(DOT3) .GE. 0.99) GO TO 210
                       ENDIF
C                 ROTATE INTACT CONCRETE'S PRINCIPAL AXES TO THE OTHER
C                 PRINCIPAL DIRECTION AND FORM A CRACK ORTHOGONAL TO IT.
C
                       HISTO(L,10,J) = PDIR(3,1)
                       HISTO(L,11,J) = PDIR(3,2)
                       HISTO(L,12,J) = PDIR(3,3)
                       HISTO(L,13,J) = PDIR(1,1)
                       HISTO(L,14,J) = PDIR(1,2)
                       HISTO(L,15,J) = PDIR(1,3)
                       HISTO(L,16,J) = PDIR(2,1)
                       HISTO(L,17,J) = PDIR(2,2)
                       HISTO(L,18,J) = PDIR(2,3)
C
                       CALL CKINIT (L, J, PROPER, ELNUM, NNEL, IOUT,
     1                     ITYPE, DTSTIF, IERROR, IFLG1, IFLG2, IFLG3,
     2                     ITCRK, ICOUP, IFLAG8)
                   ENDIF
               ENDIF
C
C  INTACT CONCRETE IN PRECRACKING PHASE AGAIN.
C
  210          CONTINUE
               HISTO(L,29,J) = 1.0
C
C   SET INTACT CONCRETE'S MODULII TO THOSE CALCULATED @
C   THE ONSET OF CRACKING.
C
C
               HISTO(L,20,J) = ES
               HISTO(L,21,J) = NUS
C
C
C    RECOVER THE STRESSES @ THE INITIATION OF CRACKING AND
C    CALCULATE THE STRAINS AT CRACKING; THESE STRAINS ARE
C    THE STRAINS AT THE INITIATION OF UNLOADING FOR INTACT
C    CONCRETE.
C
C
               SIGM1F = HISTO(L,34,J)
               SIGM2F = HISTO(L,35,J)
               SIGM3F = HISTO(L,36,J)
C
               EPS1 = (SIGM1F-NUS*SIGM2F-NUS*SIGM3F)/ES
               EPS2 = (SIGM2F-NUS*SIGM1F-NUS*SIGM3F)/ES
               EPS3 = (SIGM3F-NUS*SIGM2F-NUS*SIGM1F)/ES
               AEG = DMAX1(DMAX1(DABS(EPS1),DABS(EPS2)),DABS(EPS3))
               IF (DABS(EPS1) .LT. AEG*1E-3) EPS1 = 0.0
               IF (DABS(EPS2) .LT. AEG*1E-3) EPS2 = 0.0
               IF (DABS(EPS3) .LT. AEG*1E-3) EPS3 = 0.0
C
               HISTO(L,31,J) = EPS1
```

```
              HISTO(L,32,J) = EPS2
              HISTO(L,33,J) = EPS3
C
C     THE CONSTITUTIVE MATRIX FOR THE POINT MUST BE REGENERATE
C
              CHKGUS = .TRUE.
C
C     STIFFNESS MATRIX FOR THE ELEMENT MUST BE RECALCULATED.
C
              NSTIFF = 1
C
          ENDIF
C
C
C     EVEN THOUGH THE MATERIAL LAW IS SATISFIED AROUND THE
C     ULTIMATE POINT THE EQUILIBRIUM MAY NOT BE.THIS IS DUE
C     TO THE FACT THAT WE FORCED THE STRESS POINT THAT WAS
C     OVERSHOOTING TO LIE ON THE FAILURE ENVELOPE FOR THE
C     TIME BEING.FURTHER ITERATIONS ARE NECESSARY TO LOCATE
C     THE STRESS POINT EXACTLY.
C
C
          CONVER = 999.
      ENDIF
      RETURN
  900           FORMAT(1X,'SIGMA1=',D12.5,'FT=',D12.5)
  910       FORMAT(/,5X,' ERROR',/,10X,'FOR THE FIRST LOAD STEP'
     1           ,' THE STRESS POINT AT GAUSS POINT',I2,1X,'OF LAYER'
     2           ,I2,1X,'LIES OUTSIDE THE FAILURE ENVELOPE FOR ELEMENT'
     3          ,/,10X,'NO.',I2,'IN SUB CON12')
      END
C
C
C


C -------------------------  P  R  E  C  O  N  -----------------------
C
C     INCLUDE(PROCESS)
      SUBROUTINE PRECON(ES,NUS,CONVER,NSTIFF,TOLER,TOLERE,
     $                CHKGUS,L,J,IOUT,OVRSHT,IELEM)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C************************************************************
      COMMON /BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON /ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      REAL*8 NUS,NUOLD,NUNEW,LOWBND
      LOGICAL CHKGUS
C
C
C               SUBROUTINE TO CHECK CONVERGENCE OF THE CONCRETE
C               CONSTITUTIVE LAW WHEN THE STRESS POINT IS IN THE
C               PRE-FAILURE STAGE.
C
C               COMPARE % DIFFERENCE FROM THE PREVIOUS VALUE.
C
```

```
      C1 = 100.*(HISTO(L,20,J)-ES)/ES
      C2 = 100.*(NUS-HISTO(L,21,J))/NUS
      IF (C1.GT.TOLERE .AND. C2.LT.TOLER) THEN
C
C
C              FOR FASTER CONVERGENCE USE THE MIN. OF THE TWO 'E'
C              VALUES; ONE DETERMINED AS THE AVERAGE OF THE CURRENT
C              AND UPDATED SECANT E AND THE OTHER DETERMINED BY
C              REDUCING THE CURRENT E BY 'TOLERE' PERCENT.
C
C
         IF (OVRSHT .EQ. 0.0) THEN
             EAVERG = (HISTO(L,20,J)+ES)/2.
             EREDUC = (1.-TOLERE/100.)*HISTO(L,20,J)
             HISTO(L,20,J) = DMIN1(EAVERG,EREDUC)
         ENDIF
C
C              POSSIBILITY OF OVERSHOOTING.REDUCE 'E' GRADUALLY
C
         IF(OVRSHT.EQ.1.0)HISTO(L,20,J)=(1.-TOLERE/100.)*HISTO(L,20,J)
C
C
C             IF THE COMPARED 'ES' WAS THE VALUE AT ULTIMATE THE
C             STRESS POINT IS STILL IN THE PRE-ULTIMATE PHASE.
C
C
         HISTO(L,29,J) = 1.
C
C             UPDATE THE CONSTITUTIVE LAW AT THIS POINT
C
         CHNGUS = .TRUE.
C
C              REGENERATE STIFFNESS MATRIX FOR THE CURRENT ELEMENT.
C
         NSTIFF = 1
C
C             CARRY OUT FURTHER ITERATIONS.
C
         CONVER = 999.
         RETURN
      ENDIF
      IF (C2 .GT. TOLER) THEN
C
C
C              IN THIS CASE THE POISSON'S RATIO IS DETERMINED
C              TO BE CHANGED.HOWEVER,TO INSURE AN OVERALL SOFTENING
C              OF THE CONSTITUTIVE MATRIX WHEN THE POISSON'S RATIO
C              IS INCREASED THE YOUNG'S MODULUS SHOULD ALSO BE DECR-
C              EASED.THIS DECREASE IS ACCOMPLISHED AS FOLLOWS :
C                         (EXPLANATION IN MY NOTES)


C
C          .... THIS IS IMPORTANT IN BIAXIAL COMPRESSION CASE WHERE
C              INCREASE IN 'NU' COULD DECREASE THE COMPRESSIVE STR-
C              AINS AND RESULT IN FALSE DETECTION OF UNLOADING IN
C              SUBSEQUENT ITERATIONS.
C
         NUOLD = HISTO(L,21,J)
         EOLD = HISTO(L,20,J)
         NUNEW = (1.+TOLER/100.)*NUOLD
C
C
C              DETERMINE THE ADJUSTED YOUNG'S MODULUS CORRESPONDG.
```

```
C               TO 'NUNEW' TO ENSURE AN OVERALL SOFTENING OF THE
C               CONSTITUTIVE RELATIONSHIP.
C
C
          EADJST = EOLD*(1.-(1.+TOLER/100.)**2.*NUOLD**2.)/((1.+TOLER/
     1         100.)*(1.-NUOLD**2.))
C
C               MAKE SURE THAT 'EADJST' DOES NOT DECREASE BELOW 'ES'.
C
          IF (EADJST .GE. ES) GO TO 100
          IF (EADJST .LT. ES) THEN
C
C
C               THE REQ'D. 'E' CORRESPONDING TO 'NUNEW' BECOMES LESS
C               THAN ITS LIMITING CASE IN THIS ITERATION.SET 'EFINAL'
C               TO ITS LIMITING CASE 'ES' AND THEN CALCULATE BACK ITS
C               CORRESPONDING POISSON'S RATIO TO ENSURE AN OVERALL
C               SOFTENING AND NOT EXCEEDING THE CURRENT REQ'D. LIMITS.
C
C
             EADJST = ES
C
C               DETERMINE THE UPPER AND LOWER BOUNDS FOR 'NUNEW'.
C
             UPBND = DSQRT(1.-ES*(1.-NUOLD**2.)/EOLD)
             CONST = EOLD*NUOLD/((1.-NUOLD**2.)*ES)
             LOWBND = -(1.0-DSQRT(1.+4.D00*CONST*CONST))/(2.0*CONST)
C           WRITE(*,*) 'LOWBND OF NU',LOWBND,'UPBND OF NU',UPBND
C
C               CHECK FOR ANY ERRORS.
C
             IF (UPBND .LT. .99*LOWBND) THEN
                 WRITE (IOUT, 900) L, J, UPBND, LOWBND, IELEM, ES, EOLD
                 DIFF = DABS(100.*(EOLD-ES)/ES)
                 IF (DIFF .LE. 5.) RETURN
C##########    STOP
             ENDIF
             IF (NUS.LE.UPBND .AND. NUS.GE.LOWBND) THEN
                 NUNEW = NUS
                 GO TO 100
             ENDIF
             IF (NUNEW.LE.UPBND .AND. NUNEW.GE.LOWBND) GO TO 100
C
C
C               FINALLY WHEN NONE OF THE ABOVE CONDITIONS ARE
C               SATISFIED ARBITRARILY SELECT 'NUNEW'.
C
C
             NUNEW = UPBND
             IF (HISTO(L,29,J) .EQ. 2.) GO TO 110
          ENDIF
  100     CONTINUE
          HISTO(L,29,J) = 1.
  110     CONTINUE
          HISTO(L,20,J) = EADJST
          HISTO(L,21,J) = NUNEW
C
C              UPDATE CONSTITUTIVE LAW AT THIS POINT.
C
          CHKGUS = .TRUE.
          NSTIFF = 1
          CONVER = 999.
      ENDIF
```

```
      RETURN
  900        FORMAT(///,1X,' ERROR IN SUB. PRECON',/,5X,
     1             'DURING CALCULATION OF THE POISSONS RATIO FOR',
     2             ' GAUSS POINT :',I2,1X,'OF LAYER :',I2,/,5X,
     3             'THE UPPER BOUND BECAME LESS THAN THE LOWER ',
     4             'BOUND',/,5X,'UPPER BOUND=',D12.5,5X,
     5             'LOWER BOUND=',D12.5,/,5X,'FOR THE CURRENT ',
     6             'ITERATION THE UTILIZED SECANT MODULUS BECAME '
     7            , 'LESS THAN ITS UPDATED VALUE.',/,5X,'FOR '
     8            , 'DIFFERENCES LESS THAN 5% THE CONVERGENCE IS '
     9            , 'ASSUMED AND THE PROCESS IS CONTINUED.',
     .            'ELNUM',I3,'ENEW',D12.5,'EOLD',D12.5)
      END
C
C


C
C ------------------------------ C O N 2 3 ----------------------
C
C     INCLUDE(PROCESS)
      SUBROUTINE CON23(L,J,CONVER,NSTIFF,SIGMA1,SIGMA2,SIGMA3,PROPER,
     $         IOUT,TOLER,CHKGUS,IELEM,IFC)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C STATUS   NUPOST   LAYPRP   ENVLPE
C
C*************************************************************
      COMMON /BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON /ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      COMMON /OTTO/ AC,BC,FK1,FK2,SIG1N,SIG2N,SIG3N,FC,AAA,BBB,FT,IFG
      DIMENSION PROPER(25)
      REAL*8 NUS,NUI,NUF,J2
      LOGICAL CHKGUS
C
C
C              SUBROUTINE TO CHECK THE MATERIAL PARAMETERS IN THE
C              POST-FAILURE STATE ( FOR NON-CRACKED CONCRETE).
C
C
C         .... DETERMINE RATIO AND STATE OF GENERAL STRESSES.
C
      ES = 0.0
      IFG = 0
      EMN = 0.0
      AC = 0.0
      BC = 0.0
      FK1 = 0.0
      FK2 = 0.0
      SIG1N = 0.0
      SIG2N = 0.0
      SIG3N = 0.0
      FT = PROPER(6)
      CALL STATUS (SIGMA1, SIGMA2, SIGMA3, STATE, J, PROPER, L, IFC)
      HISTO(L,25,J) = STATE
      IF (HISTO(L,29,J) .EQ. 2.) THEN
          IF (CRACK(L,1,J) .GE. 1.) THEN
              IF (STATE.EQ.100. .OR. STATE.EQ.110. .OR. STATE.EQ.111.
```

```
     1             .OR. STATE.EQ.101.) THEN
C
C
C       THE INTACT CONCRETE IN A CRACKED POINT IS IN THE STATE
C       OF EITHER UNI,BI,TRI AXIAL TENSION, OR TEN-COMP
C       ( CRACKING ZONE ).THE POST FAILURE BEHAVIOR IS TAKEN
C       CARE OF THROUGH INTRODUCING CRACKS AND THE INTACT CONC.
C       IS ASSUMED TO BE IN THE PRE-CRACKING STAGE.
C
C
                 HISTO(L,29,J) = 1.
                 GO TO 100
             ENDIF
         ENDIF
C
C
C        IN PREVIOUS ITERATION THE STRESS POINT WAS ON THE
C        FAILURE ENVELOPE.SINCE THE STRAINS ARE INCREASING
C        THE POINT MUST NOW BE IN THE POST-FAILURE STAGE.
C        DECREASE THE PREVIOUS SECANT 'E' BY TOLER% AND
C        DETERMINE THE CORRESPONDING SECANT 'NU'.
C
C
         HISTO(L,20,J) = (1.-TOLER/100.0)*HISTO(L,20,J)
         CALL NUPOST (L, J, HISTO(L,20,J),NUS,TOLER,IOUT)
C
C             NOW THE POINT IS IN THE POST-FAILURE SEGMENT...
C
         HISTO(L,29,J) = 3.
         HISTO(L,21,J) = NUS
C
C              UPDATE THE MATERIAL LAW AT THIS POINT.
C
  100    CONTINUE
         CHKGUS = .TRUE.
         NSTIFF = 1
         CONVER = 999.
         RETURN
     ENDIF
C
C              RETRIEVE PRINCIPAL STRESSES AT FAILURE
C
     SIGM1F = HISTO(L,34,J)
     SIGM2F = HISTO(L,35,J)
     SIGM3F = HISTO(L,36,J)
C
C                  CHECK FOR OVERSHOOTING
C
     IF (STATE .EQ. (-100.)) THEN
C
C               STATE OF UNIAXIAL COMPRESSION
C
         IF (DABS(SIGMA3) .LE. DABS(SIGM3F)) GO TO 110
     ENDIF
     IF (DABS(SIGMA1).GT.DABS(SIGM1F) .OR. DABS(SIGMA2).GT.DABS(SIGM2F)
    1     .OR. DABS(SIGMA3).GT.DABS(SIGM3F)) THEN
C
C
C               AT LEAST ONE PRINCIPAL STRESS HAS EXCEEDED ITS
C               CORRESPONDING VALUE AT ULTIMATE.
C
C
C               REGARDLESS OF THE CURRENT ALLOWABLE TOLERANCE REDUCE
```

```
C               THE CURRENT MODULUS VERY SLOWLY TO AVOID EXCESSIVE
C               SOFTENING AROUND THE ULTIMATE POINT.THIS WILL ELIMI-
C               NATE THE POSSIBILITY OF DIVERGENCE.
C
C                   USE ARBITRARY DECREMENTS OF 1%
C
          HISTO(L,20,J) = .990*HISTO(L,20,J)
C
C
          CALL NUPOST (L, J, HISTO(L,20,J),NUS,TOLER,IOUT)
          HISTO(L,21,J) = NUS
C
C               UPDATE THE MATERIAL LAW AT THIS POINT.
C
          CHKGUS = .TRUE.
C
C               NEW ELEMENT STIFFNESS MATRIX MUST BE GENERATED
C


          NSTIFF = 1
C
C               FURTHER ITERATION MUST BE CARRIED OUT.
C
          CONVER = 999.
          RETURN
      ENDIF
C
C               RETRIEVE THE LAYER PROPERTIES
C
  110 CONTINUE
      CALL LAYPRP(PROPER,J,FC,FT,BETHAA,EPSC,EI,NUI,NUF,DPOST,CSIGMA)
C
C               EC = SECANT MODULUS AT MAX. UNIAXIAL COMPRESSIVE STS.
C                A = RATIO OF INITIAL MODULUS TO 'EC'.
C
C
      IF (CRACK(L,1,J) .GE. 1.0) THEN
          IF (IFC .EQ. 2) THEN
              IF (HISTO(L,41,J) .LT. PROPER(7)) THEN
                  FC = PROPER(5)*(1.0-0.2*(HISTO(L,41,J)/PROPER(7)))
                  EPSC = EPSC*(1.0+0.10*(HISTO(L,41,J)/PROPER(7)))
                  EPSC = DMAX1(PROPER(7),EPSC)
              ELSE
                  FC = 0.8*FC
                  EPSC = 1.10*EPSC
              ENDIF
          ENDIF
      ENDIF
      FC = DMIN1(HISTO(L,58,J),FC)
      HISTO(L,58,J) = FC
C
      EC = FC/EPSC
      A = EI/EC
C
C               DETERMINE FINAL VALUES OF THE PRINCIPAL STRESSES AT
C               THE TERMINATION OF THE DESCENDING PORTION OF THE
C               STRESS-STRAIN CURVE.
C
C
      FISIG1 = CSIGMA*SIGM1F
      FISIG2 = CSIGMA*SIGM2F
      FISIG3 = CSIGMA*SIGM3F
C
```

```
C
C HAVING 'SIGMA1' , 'SIGMA2' AND 'SIGMA3' DETERMINE THE CORRESPONDING
C SECANT VALUES IN THE POST-FAILURE STAGE.
C
C
      IF(STATE.EQ.(-100.).OR.STATE.EQ.(-110.).OR.STATE.EQ.(-111.))THEN
C
C
C              STATE OF UNIAXIAL OR BI/TRI AXIAL COMPRESSION.
C
C               CHECK THE POSSIBILITY OF CRUSHING (COMPLETE LOSS IN
C               STIFFNESS).
C
          IF (STATE .EQ. (-100.)) THEN
              IF (DABS(SIGMA3) .GE. DABS(FISIG3)) GO TO 120
          ENDIF
          IF (DABS(SIGMA1).LT.DABS(FISIG1) .OR. DABS(SIGMA2).LT.DABS(
     1        FISIG2) .OR. DABS(SIGMA3).LT.DABS(FISIG3)) THEN
C
C               SET MODULUS OF ELASTICITY AND THE POISSON'S RATIO
C               TO VERY SMALL NUMBERS ( ELIMINATE STIFFNESS )
C
              HISTO(L,20,J) = PROPER(1)/1000.0
              HISTO(L,21,J) = 0.001
C
C               SET THE STATE OF THE POINT TO CRUSHING.
C
              HISTO(L,29,J) = 4.
C
C
C               EVEN IF THERE WERE SOME CRACKS AT THIS POINT THEY
C               CANT EXIST AFTER THE POINT IS CRUSHED.
C
C
              CRACK(L,1,J) = 0.0
              CHKGUS = .TRUE.
              NSTIFF = 1
              CONVER = 999.
              RETURN
          ENDIF
C
C              DETERMINE THE FAILURE VALUE OF SIGMA3 ,'SIGFL3',
C              CORRESPONDING TO SIGMA1 AND SIGMA2
C
C
  120     CONTINUE
          IF (STATE .EQ. (-100.)) THEN
              SIGMA1 = 0.0
              SIGMA2 = 0.0
          ENDIF
          IFG = 1
          AAA = -1000.0*FC
          BBB = 0.0
          CALL ENVLPE (SIGMA1, SIGMA2, SIGMA3, PROPER, BETHA, SIGFL3,
     1        IOUT, STATE, SIGFL1, SIGFL2)
C
C
C       CALCULATE THE SECOND INVARIANT OF THE DEVIATORIC STRESSES
C       CORRESPONDING TO 'SIGMA1' & 'SIGMA2' SIGFL3
C
          J2 = (SIGMA1*SIGMA1+SIGMA2*SIGMA2+SIGFL3*SIGFL3-SIGMA1*SIGMA2-
     1        SIGMA1*SIGFL3-SIGMA2*SIGFL3)/3.0D00
          X = DSQRT(J2)/FC - 1.D00/DSQRT(3.0D00)
```

```
      EF = EC/(1.+4.*(A-1.)*X)
C
C           CALCULATION OF THE SECANT YOUNG'S MODULUS.
C
      ES1 = EI/2. - BETHA*(EI/2.-EF)
      ES2 = DSQRT(ES1*ES1+(EF*EF)*BETHA*(DPOST*(1.-BETHA)-1.))
      ES = ES1 - ES2
C
C           DETERMINE THE CORRESPONDING POISSON'S RATIO.
C
      CALL NUPOST (L, J, ES, NUS, TOLER, IOUT)
    ENDIF
    IF (STATE .EQ. (-101.)) THEN
C
C           STATE OF COMPRESSION-TENSION (CRUSHING ZONE).
C
C            CHECK POSSIBILITY OF CRUSHING.
C
      IF (DABS(SIGMA3) .LE. DABS(FISIG3)) THEN
C
C            ELIMINATE STIFFNESS OF THE CURRENT POINT.
C            SET THE ELASTIC MODULI TO SMALL NUMBERS TO
C            AVOID NUMERICAL PROBLEMS.
C
        HISTO(L,20,J) = PROPER(1)/1000.0
        HISTO(L,21,J) = 0.001
C
C            SET THE STATE OF THE POINT TO CRUSHING.
C
        HISTO(L,29,J) = 4.
C
C            NO CRACKS CAN EXIST AT THIS POINT ANY MORE.
C
        CRACK(L,1,J) = 0.0
        CHKGUS = .TRUE.
        NSTIFF = 1
        CONVER = 999.
        RETURN
      ENDIF
      IFG = 1
      AAA = -1000.0*FC
      BBB = 0.0
      CALL ENVLPE (SIGMA1, SIGMA2, SIGMA3, PROPER, BETHA, SIGFL3,
   1    IOUT, STATE, SIGFL1, SIGFL2)
      EF = EC
C
C           CALCULATION OF THE SECANT YOUNG'S MODULUS.
C
      ES1 = EI/2. - BETHA*(EI/2.-EF)
      ES2 = DSQRT(ES1*ES1+(EF*EF)*BETHA*(DPOST*(1.-BETHA)-1.))
      EMN = ES1 - ES2
C
C
C           RETRIEVE THE NECESSARY PARAMETERS THAT WERE CALCULATED
C           AND STORED AT THE ONSET OF FAILURE.
C
C
      EA = HISTO(L,26,J)
      EM = HISTO(L,56,J)
      BETHAF = HISTO(L,55,J)
      ES = (BETHA*EMN*EA*EM)/(BETHA*EA*EM+BETHAF*EMN*(EM-EA))
C
C           DETERMINE THE CORRESPONDING SECANT 'NU'.
```

```
C
          CALL NUPOST (L, J, ES, NUS, TOLER, IOUT)
      ENDIF
C
C
C                        ---------------------
C                        * CHECK CONVERGENCE *
C                        ---------------------
C          DO NOT OVERSOFTEN:
C                IF THE UPDATED SECANT MODULUS IS SMALLER THAN ITS
C                CORRESPONDING VALUE IN THE CURRENT ITERATION THEN
C                USE THE CURRENT VALUE AND ASSUME CONVERGENCE 3
C
C
      IF (ES .LE. HISTO(L,20,J)) THEN
          WRITE (IOUT, 900) L, J, IELEM, ES, HISTO(L,20,J)
          RETURN
      ENDIF
C
C
C               COMPARE % DIFFERENCE BETWEEN THE VALUE USE IN THE
C               CURRENT ITERATION AND THE UPDATED VALUE OBTAINED
C               BASED ON
C
C
      C = 100.*(ES-HISTO(L,20,J))/ES
      IF (C .GT. TOLER) THEN
          HISTO(L,20,J) = (1.-TOLER/100.)*HISTO(L,20,J)
C
C               FIND THE CORRESPONDING POISSON'S RATIO.
C
          CALL NUPOST (L, J, HISTO(L,20,J),NUS,TOLER,IOUT)
          HISTO(L,21,J) = NUS
C
C               UPDATE THE MATERIAL LAW.
C
          CHKGUS = .TRUE.
          NSTIFF = 1
          CONVER = 999.
      ENDIF
      RETURN
  900    FORMAT(//,2X,' WARNING',/,5X,'THE CALCULATED POST-'
     1,'FAILURE SECANT MODULUS AT G. POINT',I2,1X,'IN LAYER',
     2I2,2X,'ELEM',I2,2X,'ES=',D12.5,/,5X,'WHICH IS LESS THAN ITS VALUE'
     3          ,'OF',1X,D12.5,1X,'USED IN THE CURRENT ITERATION 3')
      END
C
C


C
C ---------------------------- N  U  P  O  S  T  ---------------------
C
C     INCLUDE(PROCESS)
      SUBROUTINE NUPOST(L,J,ES,NUS,TOLER,IOUT)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
```

```
C************************************************************
      COMMON /BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON /ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      REAL*8 NU1,NUS
C
C
C             SUBROUTINE TO DETERMINE THE POISSON'S RATIO IN THE
C             POST-FAILURE STATE.
C
C
C        ..... RETRIEVE THE SECANT YOUNG AND THE BULK MODULI AT FAILURE
C
C
      ESFLR = HISTO(L,26,J)
      BULKMD = HISTO(L,28,J)
C
C
C             FOR EACH 5% DECREASE IN 'ESFLR' THE POISSON'S RATIO IS
C             INCREASED TO  NUS=1.005*NU1 IN WHICH;
C
C                BULKMD=E(POST-FAILURE)/3(1-2*NU1)
C
C             THIS HAS THE EFFECT OF INCREASING THE BULK MODULUS AT
C             FAILURE GRADUALLY TO ACCOUNT FOR THE VOLUMETRIC INCREASE
C             THAT TAKES PLACE IN THE POST-FAILURE PHASE.
C
C
      C = (ESFLR-ES)/(.05*ESFLR)
      NU1 = ES/(-6.*BULKMD) + .50
      NUS = NU1*(1.+C*.005)
C
C             LIMIT POISSON'S RATIO TO 0.45
C
      IF (NUS .GT. 0.45) NUS = 0.45
      RETURN
      END
C
C
C
C ____________________________ U N R E L D _____________________
C
C     INCLUDE(PROCESS)
      SUBROUTINE UNRELD(EPS1,EPS2,EPS3,PROPER,CHKGUS,NSTIFF,CONVER,L,J)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C************************************************************
      COMMON /BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON /ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      DIMENSION PROPER(25)
      LOGICAL CHKGUS
      IF (DABS(EPS1).LT..999*DABS(HISTO(L,22,J)) .OR. DABS(EPS2).LT..999
     1    *DABS(HISTO(L,23,J)) .OR. DABS(EPS3).LT..999*DABS(HISTO(L,24,J
     2    ))) THEN
C
```

```
C
C                   ------------------------------
C                   *
C                   * U N L O A D I N G *
C                   *                            *
C                   ------------------------------
C
C
      HISTO(L,30,J) = 2.
C
C
C            CHECK TO SEE IF UNLOADING RESULTS IN COMPRESSIVE
C            STRAINS BECOMING TENSILE STRAINS OR VICE VERSA.
C
C
      VAR1 = EPS1*HISTO(L,22,J)
      VAR2 = EPS2*HISTO(L,23,J)
      VAR3 = EPS3*HISTO(L,24,J)
      IF (VAR1.LT.0.0 .OR. VAR2.LT.0.0 .OR. VAR3.LT.0.0) THEN
C
C
C            IF THE UNLOADING PROCESS RESULTS IN CHANGE OF THE
C            SIGN OF THE STRAIN THEN FORCE THE CURRENT STRAIN UNLOAD
C            UP TO ZERO AND THEN FOLLOW THE STRESS-STRAIN CURVE ON
C            ITS VIRGIN PATH(NOT SECANT UNLOADING E).
C
C
         IF (VAR1 .LT. 0.0) THEN
             HISTO(L,31,J) = 0.0
             HISTO(L,22,J) = 0.0
         ENDIF
         IF (VAR2 .LT. 0.0) THEN
             HISTO(L,32,J) = 0.0
             HISTO(L,23,J) = 0.0
         ENDIF
         IF (VAR3 .LT. 0.0) THEN
             HISTO(L,33,J) = 0.0
             HISTO(L,24,J) = 0.0
         ENDIF
         HISTO(L,30,J) = 1.0
         HISTO(L,29,J) = 1.0
C
C
C            SET MODULUS OF ELASTICITY AND THE POISSON'S RATIO
C            TO THEIR INITIAL VALUES.
C
C
         HISTO(L,20,J) = PROPER(1)
         HISTO(L,21,J) = PROPER(2)
C
C
C            GENERATE NEW CONSTITUTIVE MATRIX , REGENERATE NEW
C            ELEMENT STIFFNESS MATRIX AND CARRY OUT ADDITIONAL
C            ITERATIONS.
C
C
         CHKGUS = .TRUE.
         NSTIFF = 1
         CONVER = 999.
         RETURN
      ENDIF
    ENDIF
C
```

```
C
C                         ------------------------------
C                         *                            *
C                         * R E L O A D I N G *
C                         *                            *
C                         ------------------------------
C
C
      IF (DABS(EPS1).GE..999*DABS(HISTO(L,22,J)) .OR. DABS(EPS2).GE..999
     1    *DABS(HISTO(L,23,J)) .OR. DABS(EPS3).GE..999*DABS(HISTO(L,24,J
     2    ))) HISTO(L,30,J) = 3.
C
C              SAVE THE CURRENT STRAINS
C
      HISTO(L,22,J) = EPS1
      HISTO(L,23,J) = EPS2
      HISTO(L,24,J) = EPS3
      RETURN
      END
C


C
C -------------------------- R O C R A K -----------------------
C
C     INCLUDE(PROCESS)
      SUBROUTINE ROCRAK(EPS1,SIGMA1,SIGMA2,SIGMA3,PROPER,L,J,ELNUM,NNEL
     $                 ,CHKGUS,NSTIFF,CONVER,IOUT,ITYPE,DTSTIF,IERROR
     $                 ,IFLG1,IFLG2,IFLG3,ITCRK,ICOUP,IFLAG8)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C CKINIT
C
C************************************************************
      INTEGER ELNUM
      COMMON/TRANS/V1(3),V2(3),V3(3)
      COMMON /BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON /ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2

      DIMENSION PROPER(25),DTSTIF(6,6)
      LOGICAL CHKGUS
C
C
C              SUBROUTINE TO CHECK THE POSSIBILITY OF INITIATING A
C              NEW CRACK IN AN ALREADY CRACKED POINT. EVEN THOUGH
C              THE MAX. PRINCIPAL STRESS IS BELOW ITS CRACKING VALUE
C              ACCORDING TO THE CURRENT STATE OF STRESS AND ITS PROX-
C              IMITY FROM THE FAILURE ENVELOPE, THE MAX. PRINCIPAL
C              STRAIN CAN EXCEED  ITS CORRESPONDING VALUE AT THE
C              INITIATION OF THE SIRST CRACK LEADING TO A NEW CRACK.
C
C
C              EPSCRK = MAX. PRINCIPAL STRAIN OF INTACT CONCRETE AT
C                       INITIATION OF THE FIRST CRACK.
C
C
      EPSCRK = HISTO(L,37,J)
      IF (EPS1 .LT. EPSCRK) RETURN
```

```
C
C              THERE IS A POSSIBILITY OF A NEW CRACK FORMATION.
C
      NCRACK = INT(CRACK(L,1,J))
C
C              ALFA = MIN. ALLOWABLE ANGLE BETWEEN ANY TWO CRACKS.
C
      ALFA = PROPER(15)*3.1415926535/180.0
      A = SIN(ALFA)
C
C
C              IF THE ANGLE BETWEEN CURRENT PRINCIPAL DIRECTION AND AN
C              NORMAL TO THE EXISTING CRACKS IS LESS THAN 'ALFA' DO NO
C              INITIATE A NEW CRACK.
C
C
      AL1 = HISTO(L,10,J)
      AM1 = HISTO(L,11,J)
      AN1 = HISTO(L,12,J)
      AL2 = HISTO(L,13,J)
      AM2 = HISTO(L,14,J)
      AN2 = HISTO(L,15,J)
      AL3 = HISTO(L,16,J)
      AM3 = HISTO(L,17,J)
      AN3 = HISTO(L,18,J)
      DO 100 ICR = 1, NCRACK
         IMP = 9*(ICR-1) + 1
         ACL = CRACK(L,137+IMP,J)
         ACM = CRACK(L,138+IMP,J)
         ACN = CRACK(L,139+IMP,J)
         DOT = AL1*ACL + AM1*ACM + AN1*ACN
         IF (DOT .GT. 1.0) DOT = 1.0
         IF (DOT .LT. (-1.0)) DOT = -1.0
         B = DSQRT(1.0D00-DOT*DOT)
         IF (B .LT. A) RETURN
C
C
C              IF THE NEW CRACK IS FORMING AT ANGLES
C              GREATER THAN 45 (DEG.) FROM THE EXISTING ONES
C              THEN THIS MEANS THAT THE OTHER PRINCIPAL DIRECTION
C              IS REACHING THE CRACKING CONDITION. THIS NEW CRACK
C              SHOULD NOT, HOWEVER, BE DEVELOPED BASED ON THE STRAIN
C              RECORDED FOR THE PREVIOUS CRACKS FOR WHICH THE ROTATION
C              OF THE SAME PRINCIPAL DIRECTION COULD BE RESPONSIBLE.
C              THIS ALSO IMPLIES THAT THE ROTATION OF THE SAME PRIN-
C              CIPAL DIRECTION IS LIMITED TO 45 (DEG.) FROM ITS
C              PREVIOUS ORIENTATION IN THE LAST ITERATION.
C
         IF (B .GT. .50*DSQRT(2.0D00)) RETURN
  100 CONTINUE
      IF (IFLAG8 .EQ. 4) THEN
         DOT3 = V3(1)*AL1 + V3(2)*AM1 + V3(3)*AN1
         IF (DABS(DOT3) .GE. 0.99) RETURN
      ENDIF
C
C              SAVE THE CURRENT PRINCIPAL STRESSES FOR INTACT CONCRETE
C
      HISTO(L,34,J) = SIGMA1
      HISTO(L,35,J) = SIGMA2
      HISTO(L,36,J) = SIGMA3
C
C              INITIATE A NEW CRACK AND GENERATE THE RELATED DATA.
C
```

```
      CALL CKINIT (L, J, PROPER, ELNUM, NNEL, IOUT, ITYPE, DTSTIF,
     1    IERROR, IFLG1, IFLG2, IFLG3, ITCRK, ICOUP, IFLAG8)
C
C                UPDATE THE CONSTITUTIVE LAW FOR THE POINT, UPDATE THE
C                CORRESPONDING ELEMENT STIFFNESS MATRIX AND CARRY OUT
C                FURTHER ITERATIONS.
C
C
      CHKGUS = .TRUE.
      NSTIFF = 1
      CONVER = 999.
      RETURN
      END
C
C


C
C ---------------------------  U  P  D  M  A  T  ----------------------
C
C     INCLUDE(PROCESS)
      SUBROUTINE UPDMAT(MAXCRK,IOUT,PROPER,MAXCK2,L,J,DTSTIF,ICOUP,
     $IFC,IFLG1,IFLG2,IFLG3,ITCRK,TEMP11,EPSX,EPSY,EPSZ,GAMAXY,GAMAYZ
     $,GAMAXZ,IELEM)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C DTRANS    DCOMPS
C
C*************************************************************
      COMMON /BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON /ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      DIMENSION PROPER(25),DCRACK(30,30),DCON(6,6),DTSTIF(6,6)
      REAL*8 NUS,PDIR(3,3),DN(3,3),TENXYZ(6,6),TEMP11(6,6)
C
C
C               SUBROUTINE TO UPDATE CONSTITUTIVE MATRIX IN THE GLOBAL
C               AXES FOR THE CURRENT GAUSS POINT 'L' IN LAYER 'J'.
C               ARRAY 'DCON' CONTAINS THE CONSTITUTIVE LAW IN THE
C               MATERIAL AXES AND WILL LATER RETURN THAT IN THE
C               GLOBAL AXES.
C
C
      DO 110 II = 1, 6
          DO 100 JJ = 1, 6
              DCON(II,JJ) = 0.0
              TENXYZ(II,JJ) = 0.0
              DN(II,JJ) = 0.0
  100     CONTINUE
  110 CONTINUE
C
      ES = HISTO(L,20,J)
      NUS = HISTO(L,21,J)
C               UPDATE ELEMENTS OF THE 'D' MATRIX IN THE MATERL. AXES
C
      CONST = ES/((1+NUS)*(1-2.0*NUS))
      DCON(1,1) = CONST*(1.0-NUS)
      DCON(1,2) = CONST*NUS
      DCON(2,1) = DCON(1,2)
```

```
      DCON(3,1) = DCON(1,2)
      DCON(1,3) = DCON(1,2)
      DCON(2,3) = DCON(1,2)
      DCON(3,2) = DCON(1,2)
      DCON(2,2) = DCON(1,1)
      DCON(3,3) = DCON(1,1)
      DCON(4,4) = ES/(2.0*(1.0+NUS))
      DCON(5,5) = DCON(4,4)
      DCON(6,6) = DCON(4,4)
      HISTO(L,1,J) = DCON(1,1)
      HISTO(L,2,J) = DCON(1,2)
      HISTO(L,3,J) = DCON(2,2)
      HISTO(L,4,J) = DCON(3,3)
      HISTO(L,5,J) = DCON(1,3)
      HISTO(L,6,J) = DCON(2,3)
      HISTO(L,7,J) = DCON(4,4)
      HISTO(L,8,J) = DCON(5,5)
      HISTO(L,9,J) = DCON(6,6)
C
C
C              IF THE POINT IS CRUSHED STIFFNESS IS COMPLETELY LOST.
C              REGISTER THE FULL ZERO CONSTITUTIVE MATRIX FOR THIS
C              POINT.
C
      IF (HISTO(L,29,J) .EQ. 4.) GO TO 190
C
C             TRANSFER THE ABOVE TO 'DCON' MATRIX.
C
C
C             ROTATE 'DCON' TO GLOBAL AXES.
C
      PDIR(1,1) = HISTO(L,10,J)
      PDIR(1,2) = HISTO(L,11,J)
      PDIR(1,3) = HISTO(L,12,J)
      PDIR(2,1) = HISTO(L,13,J)
      PDIR(2,2) = HISTO(L,14,J)
      PDIR(2,3) = HISTO(L,15,J)
      PDIR(3,1) = HISTO(L,16,J)
      PDIR(3,2) = HISTO(L,17,J)
      PDIR(3,3) = HISTO(L,18,J)


      CALL DTRANS (DCON, PDIR, IOUT)
C
C             IF THERE ARE CRACKS AT THIS POINT
C
      IF (CRACK(L,1,J) .GE. 1.) THEN
          CALL DCOMPS (DCON, IOUT, MAXCRK, MAXCR2, DCRACK, L, J, IELEM)
C
          IF (ICOUP .EQ. 0) THEN
              IF (IFLG1.EQ.1 .AND. IFLG2.EQ.1 .AND. IFLG3.EQ.1) THEN
                  IMG = 0
                  DO 160 ISP = 1, 3
                      ISPG = TENSTF(L,3+ISP,J)
                      IF (ISPG.GT.0 .AND. TENSTF(L,ISP,J).EQ.1.0) THEN
                          ISMG = (ISPG-1)*9 + 1
                          IMG = IMG + 1
C                        IF(IFC.EQ.2) THEN
                          IF (IMG .EQ. 1) THEN
                              CRACK(L,87+ITCRK,J) = PROPER(1)/1000.0
C
C ... TENSION STIFFENING (STEEL) DIRECTION W.R.T TO 'GLOBAL'
C
                              DN(1,1) = TENSTF(L,ISMG+30,J)
```

```
            DN(1,2) = TENSTF(L,ISMG+31,J)
            DN(1,3) = TENSTF(L,ISMG+32,J)
            DN(2,1) = TENSTF(L,ISMG+33,J)
            DN(2,2) = TENSTF(L,ISMG+34,J)
            DN(2,3) = TENSTF(L,ISMG+35,J)
            DN(3,1) = TENSTF(L,ISMG+36,J)
            DN(3,2) = TENSTF(L,ISMG+37,J)
            DN(3,3) = TENSTF(L,ISMG+38,J)
            DO 130 IM = 1, 6
                DO 120 IN = 1, 6
                TENXYZ(IM,IN) = 0.0
 120            CONTINUE
 130        CONTINUE
C*******###
            ISPG2 = 0
            ISPG3 = 0
            IF (ISPG .EQ. 1) THEN
                ISPG1 = ISPG
                ISPG2 = 2
                ISPG3 = 3
            ELSE IF (ISPG .EQ. 2) THEN
                ISPG1 = ISPG
                ISPG2 = 3
                ISPG3 = 1
            ELSE IF (ISPG .EQ. 3) THEN
                ISPG1 = ISPG
                ISPG2 = 1
                ISPG3 = 2
            ENDIF
            ISP2 = 0
            ISP3 = 0
            IF (TENSTF(L,4,J) .EQ. ISPG2) ISP2 = 1
            IF (TENSTF(L,5,J) .EQ. ISPG2) ISP2 = 2
            IF (TENSTF(L,6,J) .EQ. ISPG2) ISP2 = 3
            IF (TENSTF(L,4,J) .EQ. ISPG3) ISP3 = 1
            IF (TENSTF(L,5,J) .EQ. ISPG3) ISP3 = 2
            IF (TENSTF(L,6,J) .EQ. ISPG3) ISP3 = 3
            EPSTL1 = EPSX*DN(1,1)*DN(1,1) + EPSY*DN(1,
     1          2)*DN(1,2) + EPSZ*DN(1,3)*DN(1,3) +
     2          GAMAXY*DN(1,1)*DN(1,2) + GAMAYZ*DN(1,2
     3          )*DN(1,3) + GAMAXZ*DN(1,1)*DN(1,3)
            EPSTL2 = EPSX*DN(2,1)*DN(2,1) + EPSY*DN(2,
     1          2)*DN(2,2) + EPSZ*DN(2,3)*DN(2,3) +
     2          GAMAXY*DN(2,1)*DN(2,2) + GAMAYZ*DN(2,2
     3          )*DN(2,3) + GAMAXZ*DN(2,1)*DN(2,3)
            EPSTL3 = EPSX*DN(3,1)*DN(3,1) + EPSY*DN(3,
     1          2)*DN(3,2) + EPSZ*DN(3,3)*DN(3,3) +
     2          GAMAXY*DN(3,1)*DN(3,2) + GAMAYZ*DN(3,2
     3          )*DN(3,3) + GAMAXZ*DN(3,1)*DN(3,3)
            CNST12 = 0.0
            CNST23 = 0.0
            CNST13 = 0.0
            IF (TENSTF(L,ISP,J).EQ.1.0 .AND. TENSTF(L,
     1          ISP2,J).EQ.1.0) THEN
                CNST12 = DSQRT(TENSTF(L,24+ISPG,J)/
     1              HISTO(L,26,J)*TENSTF(L,24+ISPG2,J)
     2              /HISTO(L,26,J))
            ELSE IF (TENSTF(L,ISP,J).EQ.1.0 .AND.
     1              TENSTF(L,ISP2,J).EQ.0.0) THEN
                CNST12 = DSQRT(TENSTF(L,24+ISPG,J)/
     1              HISTO(L,26,J)*HISTO(L,20,J)/HISTO(
     2              L,26,J))
                IF (EPSTL2 .GT. TENSTF(L,9+ISPG,J))
```

```
     1              CNST12 = 0.0
              ELSE IF (TENSTF(L,ISP,J).EQ.0.0 .AND.
     1              TENSTF(L,ISP2,J).EQ.1.0) THEN
                 CNST12 = DSQRT(TENSTF(L,24+ISPG2,J)/
     1              HISTO(L,26,J)*HISTO(L,20,J)/HISTO(
     2              L,26,J))
                 IF (EPSTL1 .GT. TENSTF(L,9+ISPG,J))
     1              CNST12 = 0.0
              ELSE IF (TENSTF(L,ISP,J).EQ.0.0 .AND.
     1              TENSTF(L,ISP2,J).EQ.0.0) THEN
                 CNST12 = DSQRT(HISTO(L,20,J)/HISTO(L,
     1              26,J)*HISTO(L,20,J)/HISTO(L,26,J))
                 IF (EPSTL2.GT.TENSTF(L,9+ISPG,J) .OR.
     1              EPSTL1.GT.TENSTF(L,9+ISPG,J))
     2              CNST12 = 0.0
              ENDIF
              IF (TENSTF(L,ISP,J).EQ.1.0 .AND. TENSTF(L,
     1           ISP3,J).EQ.1.0) THEN
                 CNST13 = DSQRT(TENSTF(L,24+ISPG,J)/
     1              HISTO(L,26,J)*TENSTF(L,24+ISPG3,J)
     2              /HISTO(L,26,J))
              ELSE IF (TENSTF(L,ISP,J).EQ.1.0 .AND.
     1              TENSTF(L,ISP3,J).EQ.0.0) THEN
                 CNST13 = DSQRT(TENSTF(L,24+ISPG,J)/
     1              HISTO(L,26,J)*HISTO(L,20,J)/HISTO(
     2              L,26,J))
                 IF (EPSTL3 .GT. TENSTF(L,9+ISPG,J))
     1              CNST13 = 0.0
              ELSE IF (TENSTF(L,ISP,J).EQ.0.0 .AND.
     1              TENSTF(L,ISP3,J).EQ.1.0) THEN
                 CNST13 = DSQRT(TENSTF(L,24+ISPG3,J)/
     1              HISTO(L,26,J)*HISTO(L,20,J)/HISTO(
     2              L,26,J))
                 IF (EPSTL1 .GT. TENSTF(L,9+ISPG,J))
     1              CNST13 = 0.0
              ELSE IF (TENSTF(L,ISP,J).EQ.0.0 .AND.
     1              TENSTF(L,ISP3,J).EQ.0.0) THEN
                 CNST13 = DSQRT(HISTO(L,20,J)/HISTO(L,
     1              26,J)*HISTO(L,20,J)/HISTO(L,26,J))
                 IF (EPSTL3.GT.TENSTF(L,9+ISPG,J) .OR.
     1              EPSTL1.GT.TENSTF(L,9+ISPG,J))
     2              CNST13 = 0.0
              ENDIF
              IF (TENSTF(L,ISP3,J).EQ.1.0 .AND. TENSTF(L
     1           ,ISP2,J).EQ.1.0) THEN
                 CNST23 = DSQRT(TENSTF(L,24+ISPG3,J)/
     1              HISTO(L,26,J)*TENSTF(L,24+ISPG2,J)
     2              /HISTO(L,26,J))
              ELSE IF (TENSTF(L,ISP3,J).EQ.1.0 .AND.
     1              TENSTF(L,ISP2,J).EQ.0.0) THEN
                 CNST23 = DSQRT(TENSTF(L,24+ISPG3,J)/
     1              HISTO(L,26,J)*HISTO(L,20,J)/HISTO(
     2              L,26,J))
                 IF (EPSTL2 .GT. TENSTF(L,9+ISPG,J))
     1              CNST23 = 0.0
              ELSE IF (TENSTF(L,ISP3,J).EQ.0.0 .AND.
     1              TENSTF(L,ISP2,J).EQ.1.0) THEN
                 CNST23 = DSQRT(TENSTF(L,24+ISPG2,J)/
     1              HISTO(L,26,J)*HISTO(L,20,J)/HISTO(
     2              L,26,J))
                 IF (EPSTL3 .GT. TENSTF(L,9+ISPG,J))
     1              CNST23 = 0.0
              ELSE IF (TENSTF(L,ISP2,J).EQ.0.0 .AND.
```

```
     1                                TENSTF(L,ISP3,J).EQ.0.0) THEN
                                  CNST23 = DSQRT(HISTO(L,20,J)/HISTO(L,
     1                                26,J)*HISTO(L,20,J)/HISTO(L,26,J))
                                  IF (EPSTL3.GT.TENSTF(L,9+ISPG,J) .OR.
     1                                EPSTL2.GT.TENSTF(L,9+ISPG,J))
     2                                CNST23 = 0.0
                              ENDIF
                              TEMP11(2,1) = CNST12*HISTO(L,26,J)*HISTO(L
     1                            ,27,J)
                              TEMP11(1,2) = CNST12*HISTO(L,26,J)*HISTO(L
     1                            ,27,J)
                              TEMP11(3,1) = CNST13*HISTO(L,26,J)*HISTO(L
     1                            ,27,J)
                              TEMP11(1,3) = CNST13*HISTO(L,26,J)*HISTO(L
     1                            ,27,J)
                              TEMP11(2,3) = CNST23*HISTO(L,26,J)*HISTO(L
     1                            ,27,J)
                              TEMP11(3,2) = CNST23*HISTO(L,26,J)*HISTO(L
     1                            ,27,J)
                              CALL DTRANS (TEMP11, DN, IOUT)
                              DO IM = 1, 6
                                  DO IN = 1, 6
                                  END DO
                              END DO
                          ENDIF
C             END IF
                      ENDIF
  160             CONTINUE
              ENDIF
          ENDIF
          IF (IFLG1.EQ.1 .AND. IFLG2.EQ.1 .AND. IFLG3.EQ.1) CALL DCOMPS
     1        (DCON, IOUT, MAXCRK, MAXCK2, DCRACK, L, J, IELEM)
          TEMP11(1,1) = 0.0
          TEMP11(2,2) = 0.0
          TEMP11(3,3) = 0.0
          DO 180 IK = 1, 6
              DO 170 IP = 1, 6
                  DCON(IK,IP)=DCON(IK,IP)+DTSTIF(IK,IP)+TEMP11(IK,IP)
  170         CONTINUE
  180     CONTINUE
      ENDIF
C
C                STORE THE GLOBAL CONSTITUTIVE MATRIX RETURNED IN 'DCON'.
C
  190 CONTINUE
      CRACK(L,2,J) = DCON(1,1)
      CRACK(L,3,J) = DCON(1,2)
      CRACK(L,4,J) = DCON(1,3)
      CRACK(L,5,J) = DCON(1,4)
      CRACK(L,6,J) = DCON(1,5)
      CRACK(L,7,J) = DCON(1,6)
      CRACK(L,8,J) = DCON(2,1)
      CRACK(L,9,J) = DCON(2,2)
      CRACK(L,10,J) = DCON(2,3)
      CRACK(L,11,J) = DCON(2,4)
      CRACK(L,12,J) = DCON(2,5)
      CRACK(L,13,J) = DCON(2,6)
      CRACK(L,14,J) = DCON(3,1)
      CRACK(L,15,J) = DCON(3,2)
      CRACK(L,16,J) = DCON(3,3)
      CRACK(L,17,J) = DCON(3,4)
      CRACK(L,18,J) = DCON(3,5)
      CRACK(L,19,J) = DCON(3,6)
```

```
      CRACK(L,20,J) = DCON(4,1)
      CRACK(L,21,J) = DCON(4,2)
      CRACK(L,22,J) = DCON(4,3)
      CRACK(L,23,J) = DCON(4,4)
      CRACK(L,24,J) = DCON(4,5)

      CRACK(L,25,J) = DCON(4,6)
      CRACK(L,26,J) = DCON(5,1)
      CRACK(L,27,J) = DCON(5,2)
      CRACK(L,28,J) = DCON(5,3)
      CRACK(L,29,J) = DCON(5,4)
      CRACK(L,30,J) = DCON(5,5)
      CRACK(L,31,J) = DCON(5,6)
      CRACK(L,32,J) = DCON(6,1)
      CRACK(L,33,J) = DCON(6,2)
      CRACK(L,34,J) = DCON(6,3)
      CRACK(L,35,J) = DCON(6,4)
      CRACK(L,36,J) = DCON(6,5)
      CRACK(L,37,J) = DCON(6,6)
      RETURN
      END
C
C


C
C ---------------------------- D C O M P S ---------------------
C
C     INCLUDE(PROCESS)
      SUBROUTINE DCOMPS(DCON,IOUT,MAXCRK,MAXCK2,DCRACK,L,J,IELEM)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C SIMUL
C
C*************************************************************
      COMMON /BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      DIMENSION DCON(6,*),DCRACK(MAXCK2,*),DN(6,30),DNDCRI(6,30),
     $          DSUBCT(6,6)
      REAL*8 N(6,30),NTRANS(30,6),NTD(30,6),NTDN(30,30)
      EQUIVALENCE (N,DNDCRI)
      NCRACK = INT(CRACK(L,1,J))
C
C              INITIALIZE THE NECESSARY ARRAYS.
C
      DO 120 II = 1, 3*NCRACK
          DO 100 JJ = 1, 6
              NTD(II,JJ) = 0.0
              NTRANS(II,JJ) = 0.0
              N(JJ,II) = 0.0
              DN(JJ,II) = 0.0
  100     CONTINUE
          DO 110 KK = 1, 3*NCRACK
              DCRACK(II,KK) = 0.0
              NTDN(II,KK) = 0.0
  110     CONTINUE
  120 CONTINUE
C
C              GENERATE TRANSFORMATION MATRIX 'N(3,2*(NO. OF CRACKS))'
```

```
C
      DO 130 I = 1, NCRACK
         K = 3*(I-1) + 1
C
C              THETA = ORIENTATION OF EACH CRACK ( COUNTER-CLOCKWISE FR
C                      THE GLOBAL X TO THE NORMAL OF THE CRACK ).
C
         ICK = 9*(I-1) + 1
         AL1 = CRACK(L,137+ICK,J)
         AM1 = CRACK(L,138+ICK,J)
         AN1 = CRACK(L,139+ICK,J)
         AL2 = CRACK(L,140+ICK,J)
         AM2 = CRACK(L,141+ICK,J)
         AN2 = CRACK(L,142+ICK,J)
         AL3 = CRACK(L,143+ICK,J)
         AM3 = CRACK(L,144+ICK,J)
         AN3 = CRACK(L,145+ICK,J)
C
         N(1,K) = AL1*AL1
         N(1,K+1) = AL1*AL2
         N(1,K+2) = AL3*AL1
         N(2,K) = AM1*AM1
         N(2,K+1) = AM1*AM2
         N(2,K+2) = AM3*AM1
         N(3,K) = AN1*AN1
         N(3,K+1) = AN1*AN2
         N(3,K+2) = AN3*AN1
         N(4,K) = 2.0*AL1*AM1
         N(4,K+1) = AL1*AM2 + AL2*AM1
         N(4,K+2) = AL3*AM1 + AL1*AM3
         N(5,K) = 2.0*AM1*AN1
         N(5,K+1) = AM1*AN2 + AM2*AN1
         N(5,K+2) = AM3*AN1 + AM1*AN3
         N(6,K) = 2.0*AN1*AL1
         N(6,K+1) = AN1*AL2 + AN2*AL1
         N(6,K+2) = AN3*AL1 + AN1*AL3
C
C              GENERATE TRANSPOSE OF 'N'.
C
         NTRANS(K,1) = N(1,K)
         NTRANS(K+1,1) = N(1,K+1)
         NTRANS(K+2,1) = N(1,K+2)
         NTRANS(K,2) = N(2,K)
         NTRANS(K+1,2) = N(2,K+1)
         NTRANS(K+2,2) = N(2,K+2)
         NTRANS(K,3) = N(3,K)
         NTRANS(K+1,3) = N(3,K+1)
         NTRANS(K+2,3) = N(3,K+2)
         NTRANS(K,4) = N(4,K)
         NTRANS(K+1,4) = N(4,K+1)
         NTRANS(K+2,4) = N(4,K+2)
         NTRANS(K,5) = N(5,K)
         NTRANS(K+1,5) = N(5,K+1)
         NTRANS(K+2,5) = N(5,K+2)
         NTRANS(K,6) = N(6,K)
         NTRANS(K+1,6) = N(6,K+1)
         NTRANS(K+2,6) = N(6,K+2)
C
C              DETERMINE D(CRACKS) MATRIX CONTAINING DIRECT AND SHEAR
C              STIFFNESSES OF EACH CRACK ON ITS DIAGONAL.
C
C              SCRACK = STIFFNESS ACROSS EACH CRACK.
C              GCRACK =      ;      ALONG    ;      ;  .
```

```
C
C
         SCRACK = CRACK(L,87+I,J)
         GCRCK1 = CRACK(L,97+I,J)
         GCRCK2 = CRACK(L,107+I,J)
         DCRACK(K,K) = SCRACK
         DCRACK(K+1,K+1) = GCRCK1
         DCRACK(K+2,K+2) = GCRCK2
  130 CONTINUE
C
C              PERFORM   NTD = N(TRANSPOSE) * D(CON.)     MULTIPLICATION
C
      DO 160 I = 1, 3*NCRACK
         DO 150 K = 1, 6
            DO 140 LL = 1, 6
               NTD(I,K) = NTD(I,K) + NTRANS(I,LL)*DCON(LL,K)
  140       CONTINUE
  150    CONTINUE
  160 CONTINUE
C
C               PERFORM    DN = D(CON.) * N       MULTIPLICATION.
C
      DO 190 I = 1, 6
         DO 180 K = 1, 3*NCRACK
            DO 170 LL = 1, 6
               DN(I,K) = DN(I,K) + DCON(I,LL)*N(LL,K)
  170       CONTINUE
  180    CONTINUE
  190 CONTINUE
C
C             PERFORM   NTDN = N(TRANS.) * D(CON) * N    MULTIPLICATION
C
      DO 220 I = 1, 3*NCRACK
         DO 210 K = 1, 3*NCRACK
            DO 200 LL = 1, 6
               NTDN(I,K) = NTDN(I,K) + NTD(I,LL)*N(LL,K)
  200       CONTINUE
  210    CONTINUE
  220 CONTINUE
      DO 240 I = 1, 3*NCRACK
         DO 230 K = 1, 3*NCRACK
            DCRACK(I,K) = DCRACK(I,K) + NTDN(I,K)
  230    CONTINUE


  240 CONTINUE
C
C               INVERT THE 'DCRACK' MATRIX.
C
C               N = NO. OF ROWS AND COLUMNS IN 'DCRACK'.
C
C
      NROCOL = 3*NCRACK
C
C               NRC = MAX. NO. OF ROWS (AND COLUMNS) ALLOWED.
C
      NRC = 3*MAXCRK
C
C               EPS = MIN. ALLOWABLE MAGNITUDE FOR A PIVOT ELEMENT.
C
      EPS = 10.0E-20
C
C               INVERSE OF 'DCRACK' WILL BE RETURNED IN 'DCRACK'.
C
```

```
C              DETER IS THE DETERMINANT OF 'DCRACK' THAT IS RETURNED
C              AS THE VALUE OF FUNCTION 'SIMUL'.
C
C
      DETER = SIMUL(NROCOL,DCRACK,EPS,NRC,IOUT)
      IF (DETER .EQ. 0.) THEN
          WRITE (IOUT, 900) L, J, IELEM
          STOP
      ENDIF
      DO 260 I = 1, 6
          DO 250 K = 1, 3*NCRACK
              DNDCRI(I,K) = 0.0
  250     CONTINUE
  260 CONTINUE
C
C             PERFORM   DNDCRI = DN * INVERSE OF DCRACK    MULTIPLICATIO
C
      DO 290 I = 1, 6
          DO 280 K = 1, 3*NCRACK
              DO 270 LL = 1, 3*NCRACK
                  DNDCRI(I,K) = DNDCRI(I,K) + DN(I,LL)*DCRACK(LL,K)
  270         CONTINUE
  280     CONTINUE
  290 CONTINUE
C
C
C              INITIALIZE MATRIX 'DSUBCT'.THIS MATRIX MUST BE SUBTRACTE
C              FROM THE INTACT CONCRETE CONSTITUTIVE MATRIX TO RESULT I
C              THE CRACKED-CONCRETE CONSTITUTIVE LAW.
C
C
      DO 310 I = 1, 6
          DO 300 K = 1, 6
              DSUBCT(I,K) = 0.0
  300     CONTINUE
  310 CONTINUE
C
C              COMPUTE    DSUBCT = DNDCRI * NTD
C
      DO 340 I = 1, 6
          DO 330 K = 1, 6
              DO 320 LL = 1, 3*NCRACK
                  DSUBCT(I,K) = DSUBCT(I,K) + DNDCRI(I,LL)*NTD(LL,K)
  320         CONTINUE
  330     CONTINUE
  340 CONTINUE
C
C              GENERATE THE CONSTITUTIVE MATRIX FOR CRACKED CONCRETE.
C
      DO 360 I = 1, 6
          DO 350 K = 1, 6
              DCON(I,K) = DCON(I,K) - DSUBCT(I,K)
  350     CONTINUE
  360 CONTINUE
      RETURN
  900     FORMAT(//,1X,' ERROR',/,5X,'DURING CONSTITUTIVE MATRIX '
     1       ,'INVERSION A NEGATIVE PIVOT WAS ENCOUNTERED','G.PT',I2,
     2       'LAYER',I2,'IELEM',I3)
      END
C
C


C
```

```
C***********************************************************
C     INCLUDE(PROCESS)
      SUBROUTINE STATUS(STRES1,STRES2,STRES3,STATE,J,PROPER,L,IFC)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C************************************************************
      COMMON /BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      DIMENSION PROPER(25)
C
C
C               SUBROUTINE TO DETERMINE THE STATE OF BIAXIAL STRESS.
C
C          ..... DEFINE INFINITY 'CINFIT'
C
      CINFIT = 10.0E+30
      FC = PROPER(5)
      FT = PROPER(6)
      IF (STRES1 .GE. STRES2) THEN
          IF (STRES1 .GE. STRES3) THEN
              SIGMA1 = STRES1
              IF (STRES2 .GE. STRES3) THEN
                  SIGMA2 = STRES2
                  SIGMA3 = STRES3
              ELSE
                  SIGMA2 = STRES3
                  SIGMA3 = STRES2
              ENDIF
          ELSE
              SIGMA1 = STRES3
              SIGMA2 = STRES1
              SIGMA3 = STRES2
          ENDIF
      ELSE IF (STRES1 .GE. STRES3) THEN
          SIGMA1 = STRES2
          SIGMA2 = STRES1
          SIGMA3 = STRES3
      ELSE
          SIGMA3 = STRES1
          IF (STRES2 .GE. STRES3) THEN
              SIGMA1 = STRES2
              SIGMA2 = STRES3
          ELSE
              SIGMA1 = STRES3
              SIGMA2 = STRES2
          ENDIF
      ENDIF
C
C
      IF (IFC.EQ.2 .AND. CRACK(L,1,J).GE.1.0) THEN
          IF (HISTO(L,41,J) .LT. PROPER(7)) THEN
              FC = FC*(1.0-0.2*(HISTO(L,41,J)/PROPER(7)))
          ELSE
              FC = FC*0.8
          ENDIF
      ENDIF
```

```
      FC = DMIN1(HISTO(L,58,J),FC)
      HISTO(L,58,J) = FC
C
C              STATE OF UNIAXIAL TENSION.
C
      IF (SIGMA2.GE.(-.0001) .AND. SIGMA2.LE..0001 .AND. SIGMA3.GE.(-
     1    .0001) .AND. SIGMA3.LE..0001 .AND. SIGMA1.GT.0.0) STATE =
     2    100.0
C
C              STATE OF BIAXIAL TENSION.
C
      IF (SIGMA1.GT.0.0 .AND. SIGMA2.GT.0.0 .AND. SIGMA3.GE.(-.0001)
     1     .AND. SIGMA3.LE..0001) STATE = 110.
C
C              STATE OF TRIAXIAL TENSION.
C
      IF(SIGMA1.GT.0.0.AND.SIGMA2.GT.0.0.AND.SIGMA3.GT.0.0)STATE=111.
C
      IF (SIGMA1.GT.0.0 .AND. SIGMA3.LT.0.0) THEN
          R = SIGMA1/SIGMA3
          RC = FT/FC
          FBD = -0.73333*RC
          IF (R.GT.(-CINFIT) .AND. R.LT.(-0.7333*RC)) THEN
C
C              STATE OF TENSION-COMPRESSION (CRACKING ZONE)
C
              STATE = 101.0
          ELSE IF (R.GE.(-0.7333*RC) .AND. R.LE.0.0) THEN
C
C              STATE OF TENSION-COMPRESSION (CRUSHING ZONE)
C
              STATE = -101.0
          ENDIF
      ENDIF
C
C              STATE OF UNIAXIAL COMPRESSION.
C
      IF (SIGMA3.LT.0.0 .AND. SIGMA2.GE.(-.0001) .AND. SIGMA2.LE..0001
     1     .AND. SIGMA1.GE.(-.0001) .AND. SIGMA1.LE..0001) STATE =
     2    -100.0
C
C              STATE OF BIAXIAL COMPRESSION.
C
      IF (SIGMA2.LT.0.0 .AND. SIGMA3.LT.0.0 .AND. SIGMA1.GE.(-.0001)
     1     .AND. SIGMA1.LE..0001) STATE = -110.0
C
C              STATE OF TRIAXIAL COMPRESSION.
C
      IF (SIGMA1.LT.0.0 .AND. SIGMA2.LT.0.0 .AND. SIGMA3.LT.0.0) STATE
     1     = -111.0
      RETURN
      END
C
C
C
C -------------------------- M A T E R L -----------------------
C
C     INCLUDE(PROCESS)
      SUBROUTINE MATERL(NGUS,NLAYER,SIGMA1,SIGMA2,SIGMA3,STATE,
     $            ES,NUS,IOUT,PROPER,BETHA,IFC,IELEM)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
```

```
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ENVLPE    TENSIL    LAYPRP    BFAILR
C
C************************************************************
      COMMON /BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON /OTTO/ AC,BC,FK1,FK2,SIG1N,SIG2N,SIG3N,FC,AAA,BBB,FT,IFG
      DIMENSION PROPER(25)
      REAL*8 NUI,NUF,NUS,J2
C
C
C              SUBROUTINE TO DETERMINE SECANT VALUES OF 'E' AND 'NU'
C              FOR THE CURRENT STATE OF STRESS, BEFORE ULTIMATE &
C              AT THE ULTIMATE.
C
C
      IFG = 0.0
      AC = 0.0
      BC = 0.0
      FK1 = 0.0
      FK2 = 0.0
      SIG1N = 0.0
      SIG2N = 0.0
      SIG3N = 0.0
      FC = PROPER(5)
      FT = PROPER(6)
      IF (CRACK(NGUS,1,NLAYER) .GE. 1.0) THEN
          IF (IFC .EQ. 2) THEN


              IF (HISTO(NGUS,41,NLAYER) .LT. PROPER(7)) THEN
                  FC = FC*(1.0-0.2*(HISTO(NGUS,41,NLAYER)/PROPER(7)))
              ELSE
                  FC = FC*0.8
              ENDIF
          ENDIF
      ENDIF
      FC = DMIN1(HISTO(NGUS,58,NLAYER),FC)
      FC = HISTO(NGUS,58,NLAYER)
      IF (STATE.EQ.100. .OR. STATE.EQ.110. .OR. STATE.EQ.111.) THEN
C
C              STATE OF UNIAXIAL, BIAXIAL OR TRIAXIAL TENSION.
C
C
C              BETHA = NONLINEARITY INDEX.
C
          IFG = 1
          AAA = -1000.0*FC
          BBB = 0.0
          CALL ENVLPE (SIGMA1, SIGMA2, SIGMA3, PROPER, BETHA, SIGM3F,
     1        IOUT, STATE, SIGM1F, SIGM2F)
C
C              CALCULATE SECANT VALUES OF 'E' AND 'NU'.
          IF (BETHA .GT. 1.0) THEN
              WRITE (*, *) 'SIGMA1', SIGMA1, 'SIGMA2', SIGMA2, 'SIGMA3'
     1            , SIGMA3
              WRITE (*, *) 'SIGMA1F', SIGM1F, 'SIGM2F', SIGM2F, 'SIGM3F'
     1            , SIGM3F
              WRITE (*, *) '1 # LAYER', NLAYER, 'GPNT', NGUS, 'ELEM',
     1            IELEM
          ENDIF
```

```
          CALL TENSIL (ES, NUS, BETHA, IOUT, PROPER, NLAYER, NGUS, IFC,
     1         SIGMA1, SIGMA2, SIGM3F, IELEM)
      ENDIF
      IF (STATE .EQ. 101) THEN
C
C             STATE OF TENSION-COMPRESSION (CRACKING ZONE)
C
C                           TOLERANCE = .0001
C
          IFG = 1
          AAA = -1000.0*FC
          BBB = 0.0
          CALL ENVLPE (SIGMA1, SIGMA2, SIGMA3, PROPER, BETHA, SIGM3F,
     1         IOUT, STATE, SIGM1F, SIGM2F)
C
C
C             CALCULATE SECANT VALUES OF 'E' AND 'NU'.
          IF (BETHA .GT. 1.0) THEN
              WRITE (*, *) 'SIGMA1', SIGMA1, 'SIGMA2', SIGMA2, 'SIGMA3'
     1            , SIGMA3
              WRITE (*, *) 'SIGMA1F', SIGM1F, 'SIGM2F', SIGM2F, 'SIGM3F'
     1            , SIGM3F
              WRITE (*, *) '2 # LAYER', NLAYER, 'GPNT', NGUS, 'ELEM',
     1            IELEM
          ENDIF
          CALL TENSIL (ES, NUS, BETHA, IOUT, PROPER, NLAYER, NGUS, IFC,
     1         SIGMA1, SIGMA2, SIGM3F, IELEM)
      ENDIF
      IF (STATE .EQ. (-101.)) THEN
C
C             STATE OF TENSION-COMPRESSION (CRUSHING ZONE)
C
          IFG = 1
          AAA = -1000.0*FC
          BBB = 0.0
          CALL ENVLPE (SIGMA1, SIGMA2, SIGMA3, PROPER, BETHA, SIGM3F,
     1         IOUT, STATE, SIGM1F, SIGM2F)
C
C
C             CALCULATE SECANT VALUES OF 'E' AND 'NU'.
          IF (BETHA .GT. 1.0) THEN
              WRITE (*, *) 'SIGMA1', SIGMA1, 'SIGMA2', SIGMA2, 'SIGMA3'
     1            , SIGMA3
              WRITE (*, *) 'SIGMA1F', SIGM1F, 'SIGM2F', SIGM2F, 'SIGM3F'
     1            , SIGM3F
              WRITE (*, *) '3 # LAYER', NLAYER, 'GPNT', NGUS, 'ELEM',
     1            IELEM
          ENDIF
          CALL TENSIL (ES, NUS, BETHA, IOUT, PROPER, NLAYER, NGUS, IFC,
     1         SIGMA1, SIGMA2, SIGM3F, IELEM)
      ENDIF
      IF(STATE.EQ.(-100.).OR.STATE.EQ.(-110.).OR.STATE.EQ.(-111.))THEN
C
C             STATE OF UNIAXIAL/BIAXIAL/TRIAXIAL  COMPRESSION.
C
C
C             CALCULATE THE FAILURE VALUE OF SIGMA2 WHEN SIGMA1 IS
C             KEPT CONSTANT.
C
          IFG = 1
          AAA = -1000.0*FC
          BBB = 0.0
          CALL ENVLPE (SIGMA1, SIGMA2, SIGMA3, PROPER, BETHA, SIGM3F,
```

```
     1        IOUT, STATE, SIGM1F, SIGM2F)
C
C
C            RETRIEVE OTHER LAYER PROPERTIES
C
        CALL LAYPRP (PROPER, NLAYER, FC, FT, BETHAA, EPSC, EI, NUI,
     1      NUF, DPOST, CSIGMA)
        IF (CRACK(NGUS,1,NLAYER) .GE. 1.0) THEN
            IF (IFC .EQ. 2) THEN
                IF (HISTO(NGUS,41,NLAYER) .LT. PROPER(7)) THEN
                    FC=FC*(1.0-0.2*(HISTO(NGUS,41,NLAYER)/PROPER(7)))
                    EPSC = EPSC*(1.0+0.10*(HISTO(NGUS,41,NLAYER)/
     1                  PROPER(7)))
                    EPSC = DMAX1(PROPER(7),EPSC)
                ELSE
                    FC = FC*0.8
                    EPSC = EPSC*1.10
                ENDIF
                FC = DMIN1(HISTO(NGUS,58,NLAYER),FC)
                FC = HISTO(NGUS,58,NLAYER)
            ENDIF
        ENDIF
        EC = FC/EPSC
        A = EI/EC
C
C
C     CALCULATE THE SECOND INVARIANT OF THE DEVIATORIC STRESS
C     CORRESPONDING TO THE STRESS STATE 'SIGMA1' 'SIGMA2' & 'SIGM3F'.
C
C
        J2 = (SIGMA1*SIGMA1+SIGMA2*SIGMA2+SIGM3F*SIGM3F-SIGMA1*SIGMA2-
     1      SIGMA2*SIGM3F-SIGM3F*SIGMA1)/3.0
        X = DSQRT(J2)/FC - 1./DSQRT(3.0D00)
C
C            EF = SECANT VALUE OF YOUNG'S MODULUS AT GENERAL
C                 STATE OF BIAXIAL COMPRESSIVE FAILURE.
C
        EF = EC/(1.+4.*(A-1.)*X)
C
C            CALCULATE SECANT VALUES OF 'E' & 'NU'.
C
        ES1 = EI/2. - BETHA*(EI/2.-EF)
        ES2 = DSQRT(DABS(ES1*ES1+(EF*EF)*BETHA*(DPOST*(1.-BETHA)-1.)))
        IF (BETHA .GT. 1.0) THEN
            WRITE (*, *) 'SIGMA1', SIGMA1, 'SIGMA2', SIGMA2, 'SIGMA3'
     1          , SIGMA3
            WRITE (*, *) 'SIGMA1F', SIGM1F, 'SIGM2F', SIGM2F, 'SIGM3F'
     1          , SIGM3F
            WRITE (*, *) '4 # LAYER', NLAYER, 'GPNT', NGUS, 'ELEM',
     1          IELEM
        ENDIF
        CALL BFAILR (BETHA, IOUT, ES1, ES2, BETHAA, NUI, NUF, ES, NUS
     1      , NGUS, NLAYER, IELEM)
    ENDIF
    RETURN
    END
C
C


C
C ____________________________  T E N S I L  ____________________

C
```

```
C     INCLUDE(PROCESS)
      SUBROUTINE TENSIL(ES,NUS,BETHA,IOUT,PROPER,NLAYER,NGUS,IFC,SIGM1F,
     $SIGM2F,SIGM3F,IELEM)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C LAYPRP    BFAILR
C
C************************************************************
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      DIMENSION PROPER(25)
      REAL*8 NUI,NUF,NUS,J2
      CALL LAYPRP (PROPER, NLAYER, FC, FT, BETHAA, EPSC, EI, NUI, NUF,
     1    DPOST, CSIGMA)
C
C
C               EF=SECANT VALUE OF YOUNG'S MODULUS AT UNIAXIAL
C                  COMPRESSIVE FAILURE.
C
C
      IF (CRACK(NGUS,1,NLAYER) .GE. 1.0) THEN
          IF (IFC .EQ. 2) THEN
              IF (HISTO(NGUS,41,NLAYER) .LT. PROPER(7)) THEN
                  FC = FC*(1.0-0.2*(HISTO(NGUS,41,NLAYER)/PROPER(7)))
                  EPSC = EPSC*(1.0+0.10*(HISTO(NGUS,41,NLAYER)/PROPER(7)
     1                ))
                  EPSC = DMAX1(PROPER(7),EPSC)
              ELSE
                  FC = FC*0.8
                  EPSC = EPSC*1.10
              ENDIF
              FC = DMAX1(HISTO(NGUS,58,NLAYER),FC)
              HISTO(NGUS,58,NLAYER) = FC
          ENDIF
      ENDIF
C
C               EF = SECANT VALUE OF YOUNG'S MODULUS AT GENERAL
C                    STATE OF BIAXIAL COMPRESSIVE FAILURE.
C
      EC = FC/EPSC
      EF = EC
C
C               CALCULATE SECANT VALUES OF 'E' & 'NU'.
C
      ES1 = EI/2. - BETHA*(EI/2.-EF)
      ES2 = DSQRT(ES1*ES1+(EF*EF)*BETHA*(DPOST*(1.-BETHA)-1.))
      CALL BFAILR (BETHA, IOUT, ES1, ES2, BETHAA, NUI, NUF, ES, NUS,
     1    NGUS, NLAYER, IELEM)
      RETURN
      END
C
C
C
C ---------------------------  L  A  Y  P  R  P  ----------------------
C
CC    INCLUDE(PROCESS)
      SUBROUTINE LAYPRP(PROPER,NLAYER,FC,FT,BETHAA,EPSC,EI,NUI,NUF,
     $                  DPOST,CSIGMA)
```

```
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C************************************************************
      DIMENSION PROPER(25)
      REAL*8 NUI,NUF
C
C
C               SUBROUTINE TO PROVIDE MATERIAL PROPERTIES FOR THE
C               CURRENT CONCRETE LAYER.
C
C
      FC = PROPER(5)
      FT = PROPER(6)
      BETHAA = PROPER(13)
      EPSC = PROPER(7)
      EI = PROPER(1)
      NUI = PROPER(2)
      NUF = PROPER(12)
      DPOST = PROPER(9)
      CSIGMA = PROPER(14)
      RETURN
      END
C
C
C
C --------------------------  B  F  A  I  L  R  ----------------------
C
C     INCLUDE(PROCESS)
      SUBROUTINE BFAILR(BETHA,IOUT,ES1,ES2,BETHAA,NUI,NUF,ES,NUS,NGUS
     $                ,LYRNO,IELEM)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C************************************************************
      COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      REAL*8 NUI,NUF,NUS
C
C
C                SUBROUTINE TO CALCULATE CURRENT SECANT E & POISSON'S
C                RATIO WHEN THE STATE OF STRESS IS INSIDE THE FAILURE
C
C
      IF (BETHA .GE. 1.01) THEN
          WRITE (IOUT, 900) BETHA, NGUS, LYRNO, IELEM
          STOP
      ENDIF
      EM = ES1 - ES2
      ES = ES1 + ES2
      HISTO(NGUS,56,LYRNO) = EM
```

```
      IF (BETHA .LE. BETHAA) NUS = NUI
      IF (BETHA .GT. BETHAA) NUS = NUF - (NUF-NUI)*DSQRT(DABS(1.0-((
     1    BETHA-BETHAA)/(1.0-BETHAA))**2.0))
      RETURN
  900    FORMAT(//,10X,'THE NONLINEARITY INDEX IS GREATER THAN 1.0 ',
     1         'WHEN THE STRESS STATE IS INSIDE THE ',
     2         'ENVELOPE 3',/,30X,'BETHA=',D12.5,'FOR G.PT',I2,'LAYER'
     3         ,I2,'ELNUM',I3)
      END
C
C
C
C ---------------------------  F  A  I  L  U  R  ---------------------
C
C     INCLUDE(PROCESS)
      SUBROUTINE FAILUR(L,LYRNO,PROPER,IOUT,ESFLR,NUSFLR,SIGMA1,SIGMA2
     $               ,SIGMA3,IFC,IELEM)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ENVLPE    MATERL
C
C************************************************************
      COMMON /BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON /ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      COMMON /OTTO/ AC,BC,FK1,FK2,SIG1N,SIG2N,SIG3N,FC,AAA,BBB,FT,IFG
      DIMENSION PROPER(25)
      REAL*8 NUSFLR
C
C
C     FOR EACH GAUSS POINT DETERMINE THE FAILURE VALUES
C     OF 'SIGMA1' , 'SIGMA2' & SIGMA3 GIVEN THE STATE OF STRESS
C     AND THE RATIO OF STRESSES.THIS RATIO IS THE CURRENT
C     ONE FOR THE STRESS POINT THAT HAS VIOLATED THE
C     FAILURE ENVELOPE OR MAY BE ON IT.
C
C
      IFG = 0.0
      AC = 0.0
      BC = 0.0
      FK1 = 0.0
      FK2 = 0.0
      SIG1N = 0.0
      SIG2N = 0.0
      SIG3N = 0.0
      STATE = HISTO(L,25,LYRNO)
      FC = PROPER(5)
      FT = PROPER(6)
      IF (CRACK(L,1,LYRNO) .GE. 1.0) THEN
          IF (IFC .EQ. 2) THEN
              IF (HISTO(L,41,LYRNO) .LT. PROPER(7)) THEN
                  FC = FC*(1.0-0.2*(HISTO(L,41,LYRNO)/PROPER(7)))
              ELSE
                  FC = FC*0.8
              ENDIF
          ENDIF
      ENDIF
      FC = DMIN1(HISTO(L,58,LYRNO),FC)
```

```
      HISTO(L,58,LYRNO) = FC
      IF (STATE .EQ. 100.) THEN
C
C              STATE OF UNIAXIAL TENSION.
C
          SIGM1F = FT
          SIGM2F = 0.0
          SIGM3F = 0.0

          GO TO 100
      ENDIF
      IF (STATE .EQ. 110.) THEN
C
C              STATE OF BIAXIAL TENSION.
C
          SIGM1F = FT
          RATIO = SIGMA2/SIGMA1
          SIGM2F = SIGM1F*RATIO
          SIGM3F = 0.0
          GO TO 100
      ENDIF
      IF (STATE .EQ. 111.) THEN
C
C              STATE OF TRIAXIAL TENSION.
C
          SIGM1F = FT
          RATIO1 = SIGMA2/SIGMA1
          RATIO2 = SIGMA3/SIGMA1
          SIGM2F = SIGM1F*RATIO1
          SIGM3F = SIGM1F*RATIO2
          GO TO 100
      ENDIF
      IF (STATE .EQ. 101) THEN
C              STATE OF TENSION-COMPRESSION (CRACKING ZONE)
C
          IFG = 2
          AAA = 0.0
          BBB = FT*1.5
C.. NOTE STRESSES SUPPLIED INVERSLY TO RECORD THE TENSILE CRACKING
C   STRESS LESS THAN OR EQUAL TO FT.
          CALL ENVLPE (SIGMA3, SIGMA2, SIGMA1, PROPER, BETHA, SIG3F,
     1        IOUT, STATE, SIG1F, SIG2F)
          SIGM1F = SIG1F
          SIGM2F = SIGM1F*SIGMA2/SIGMA1
          SIGM3F = SIGM1F*SIGMA3/SIGMA1
      ENDIF
      IF (STATE .EQ. (-101.)) THEN
C
C              STATE OF TENSION-COMPRESSION (CRUSHING ZONE)
C
          IFG = 2
          AAA = -1000.0*FC
          BBB = 0.0
          CALL ENVLPE (SIGMA1, SIGMA2, SIGMA3, PROPER, BETHA, SIG3F,
     1        IOUT, STATE, SIGM1F, SIGM2F)
          SIGM3F = SIG3F
          SIGM2F = SIGM3F*SIGMA2/SIGMA3

          SIGM1F = SIGM3F*SIGMA1/SIGMA3
      ENDIF
      IF(STATE.EQ.(-100.).OR.STATE.EQ.(-110.).OR.STATE.EQ.(-111.))THEN
C
C              STATE OF UNIAXIAL OR BI/TRI AXIAL COMPRESSION.
```

```
C
         AAA = -1000.0*FC
         BBB = 0.0
         IFG = 2
         CALL ENVLPE (SIGMA1, SIGMA2, SIGMA3, PROPER, BETHA, SIG3F,
     1         IOUT, STATE, SIGM1F, SIGM2F)
         SIGM3F = SIG3F
         SIGM2F = SIGM3F*(SIGMA2/SIGMA3)
         SIGM1F = SIGM3F*(SIGMA1/SIGMA3)
      ENDIF
C
C              DETERMINE MATERIAL PARAMETERS AT FAILURE.
C
  100 CONTINUE
      CALL MATERL (L, LYRNO, SIGM1F, SIGM2F, SIGM3F, STATE, ESFLR,
     1    NUSFLR, IOUT, PROPER, BETHAF, IFC, IELEM)
C
C              SAVE THESE VALUES FOR EACH GAUSS POINT.
C
      HISTO(L,26,LYRNO) = ESFLR
      HISTO(L,27,LYRNO) = NUSFLR
C
C              DETERMINE BULK MODULUS AT FAILURE.
C
      BULKMD = ESFLR/(3.*(1.-2.*NUSFLR))
      HISTO(L,28,LYRNO) = BULKMD
      HISTO(L,55,LYRNO) = BETHAF
C
C              SAVE PRINCIPAL STRESSES AT ULTIMATE.
C
      HISTO(L,34,LYRNO) = SIGM1F
      HISTO(L,35,LYRNO) = SIGM2F
      HISTO(L,36,LYRNO) = SIGM3F
      RETURN
      END
C
C
C
C************************************************************
C     INCLUDE(PROCESS)
      SUBROUTINE ENVLPE(SIGMA1,SIGMA2,SIGMA3,PROPER,BETHA
     $                 ,SIGM3F,IOUT,STATE,SIGM1F,SIGM2F)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ROOT      DSVRGP
C
C************************************************************
      COMMON /OTTO/ AC,BC,FK1,FK2,SIG1N,SIG2N,SIG3N,FCP,A,B,FT,IFG
      EXTERNAL FVAL
      DIMENSION BOUNDS(100),WS(100),WS1(100)
     $          ,ALDB(10),AUDB(10),IPERM1(10),IPERM2(10),PROPER(25)
C
C   GIVEN THE STATE OF STRESS IT WILL EVALUATE THE
C   NONLINEAR INDEX ASSOCIATED WITH THE FAILURE ENVELOPE.
C
      PI = 3.141592654
      AC = PROPER(18)
      BC = PROPER(19)
```

```
      FK1 = PROPER(20)
      FK2 = PROPER(21)
C     FCP=PROPER(5)
C
      BETHA = -1.0
      SIG1N = 0.0
      SIG2N = 0.0
      SIG3N = 0.0
      SIGM1F = 0.0
      SIGM2F = 0.0
      SIGM3F = 0.0
      IF (IFG .EQ. 1) THEN
          IF (SIGMA1 .GE. 0.0) THEN
              SIG1N = 0.0
              SIG2N = SIGMA2 - SIGMA1
              SIG3N = SIGMA3 - SIGMA1
          ELSE
              SIG1N = SIGMA1
              SIG2N = SIGMA2
              SIG3N = SIGMA3
          ENDIF
      ELSE IF (IFG .EQ. 2) THEN
          SIG1N = SIGMA1
          SIG2N = SIGMA2
          SIG3N = SIGMA3
      ENDIF
C
      L = 100
      NQ = 3
      T2 = 5
      E = 0.00001
C
C     FIND THE ROOTS
C
      CALL ROOT (FVAL, A, B, NQ, T2, E, BOUNDS, NROOT, WS, WS1, L)
C
C FIND THE FAILURE VALUE OF STRESS FOR GIVEN SIGMA 1 AND 2
C
      IF (NROOT .GT. 10) WRITE (IOUT, *) 'NO#.ROOTS EXCEEDED MAX ENVLPE'
      IF (NROOT .EQ. 0) WRITE (*, *) ' NO ROOTS IN BOUND A', A, 'B'
     1    , B, '<<<<<'
      DO 100 IM = 1, NROOT
          I1 = 2*IM - 1
          I2 = 2*IM
          ALDB(IM) = BOUNDS(I1)
          AUDB(IM) = BOUNDS(I2)
          IPERM1(IM) = IM
          IPERM2(IM) = IM
  100 CONTINUE
      IF (NROOT .GT. 0) THEN
          CALL DSVRGP (NROOT, ALDB, ALDB, IPERM1)
          CALL DSVRGP (NROOT, AUDB, AUDB, IPERM2)
      ENDIF
      DO 110 IM = 1, NROOT
          IF (IPERM1(IM) .NE. IPERM2(IM)) WRITE (NROOT, *)
     1        'PERMUTATION MISTAKE IN ENVLPE'
          ALDB(IM) = (ALDB(IM)+AUDB(IM))/2.0
  110 CONTINUE
      IF (STATE.EQ.101 .AND. IFG.EQ.2) GO TO 140
      DO 120 IPN = NROOT, 1, -1
          IF (DABS(SIG3N) .LE. DABS(ALDB(IPN))) THEN
              SIGM3F = ALDB(IPN)
              GO TO 130
```

```
          ENDIF
  120 CONTINUE
      IF (SIGM3F .EQ. 0.0) THEN
          WRITE (*, *) '   FAILURE VALUE NOT FOUND  IFG', IFG
          WRITE (*, *) 'SIG1N', SIG1N, 'SIG2N', SIG2N, 'SIG3N', SIG3N
          SIGM3F = -1.16*FCP
          WRITE (*, *) 'FC', FCP, 'SIG3N', SIG3N, 'SIGM3F', SIGM3F
      ENDIF
C
  130 CONTINUE
      IF (IFG .EQ. 1) BETHA = DABS(SIG3N)/DABS(SIGM3F)
      IF (BETHA .GT. 1.0) WRITE (*, 900)
C

  140 CONTINUE
      IF (STATE.EQ.101 .AND. IFG.EQ.2) THEN
          DO 150 IMN = 1, NROOT
              IF (ALDB(IMN).GT.0.0 .AND. ALDB(IMN).LE.FT) THEN
                  SIGM1F = ALDB(IMN)
                  RETURN
              ENDIF
  150     CONTINUE
          IF (SIGM1F .EQ. 0.0) THEN
              WRITE (*, *) '      NO FAILURE VALUE OBTAINED   IFG', IFG
              WRITE (*, *) 'SIG1N', SIG1N, 'SIG2N', SIG2N, 'SIG3N',
     1            SIG3N
              WRITE (*, *) 'SETTING SIGM3F TO BE SIG3N-TENSION-CUTOFF'
              WRITE (*, *) 'FT', FT, 'FVAL', ALDB(NROOT)
              SIGM1F = FT
          ENDIF
      ENDIF
C
      RETURN
  900        FORMAT(1x,'LOAD INCREMENT SIZE TOO LARGE REFINE STEP SIZE',
     1              'CHECK VALUES OF GF AND TENSION STIFFENING LAYER ALSO'
     2              'THESE QUANTITIES MAY BE SMALL LEADING TO FALSE CRUSHING')
      END


C ****************************************************************
C     INCLUDE(PROCESS)
      SUBROUTINE OTOSEN(FCP,FCT,PROPER,IOUT)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C***************************************************************
      DIMENSION PROPER(25)
      AK = FCT/FCP
C
C X AND Y ARE DEFINED BASED ON BALMER(1929) TRIAXIAL TEST DATA
C  (X,Y) =(-5/DSQRT3,4/DSQRT3) FOR SCHICKERT DATA USE 3.28 FOR 4
C
      X = -50.0/FCP
      Y = 43.61/FCP
C      X = -5.0 / DSQRT(3.0D00)
C      Y = 3.28 / DSQRT(3.0D00)
C
```

```
C   BI-AXIAL STRENGTH DATA 1.16 FOR KUPFER DATA AND 1.21 FOR
C   SCHICKERT DATA.
C
      SIGBC = 1.16
C       SIGBC = 1.21
C
      H = -(DSQRT(2.0D00)*X+Y)/(Y/DSQRT(2.0D00)-1./3.)
      B = (DSQRT(2.0D00)-(3.*Y)/(AK*SIGBC))/(H-(9.0*Y)/(SIGBC-AK))
      A = (H*B-DSQRT(2.0D00))/Y
C
      ALAMBC = (1.0-H/(3.*Y))*DSQRT(3.0D00)*B + DSQRT(3.0D00) + DSQRT(
     1    2.0D00)/(DSQRT(3.0D00)*Y)
C
      ALAMBT = (2.*DSQRT(3.0D00)-(SIGBC*H)/(DSQRT(3.0D00)*Y))*B + (
     1    DSQRT(2.0D00)*SIGBC)/(DSQRT(3.0D00)*Y) + DSQRT(3.0D00)/SIGBC
C
      FK2 = DCOS(3.0*DATAN((2.0*ALAMBC/ALAMBT-1.0)/DSQRT(3.0D00)))
      FK1 = ALAMBT/DCOS((1.0D00/3.0D00)*DACOS(FK2))
C
      PROPER(18) = A
      PROPER(19) = B
      PROPER(20) = FK1
      PROPER(21) = FK2
C
      RETURN
      END
C                         ***************************
C                         *  ROOT ISOLATION PROGRAM *
C ***********************                          **********************
C    PROGRAM TO ISOLATE THE ROOTS FOR A TRANSCENDENTAL EQUATION       *
C **********************************************************************
C
C     INCLUDE(PROCESS)
      SUBROUTINE ROOT(FVAL,A,B,NQ,T2,E,BOUNDS,NROOT,WS,WS1,L)
C
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C FVAL
C
C************************************************************
      EXTERNAL FVAL
      DIMENSION BOUNDS(*),WS(1),WS1(1),STORE(47),STORE1(47)
      INTEGER DIGITS
      DOUBLE PRECISION SUM1,SUM2
      DATA EPS2,EPS3,DIGITS/1.E-6,1.E-7,6/
      DATA ZERO/0.0/,TWO/2.0/,EPS1/0.2/,ISIZE/47/,LIMIT/10/
      DATA ASIZE/47.0/
C
C
C.. INITIALIZE
C
      XOLD = 0.0
      K = 0
      KD = 0
      FX = 0
      H = 0
      XVOLD = 0.0
      SUM2 = 0.0
```

```
      SUM1 = 0.0
      XMOLD = 0.0
      NR = 0
      FOLD = 0.0
C
      C = B + DABS(B)
      DO 100 M = 1, ISIZE
         STORE1(M) = C
  100 CONTINUE
      NROOT = 0
      J = 0
C
      BOUNDS(1) = A
      XSTART = A + (B-A)*0.2318
      BOUNDS(2) = XSTART
      XEND = B
      I = 1
  110 CONTINUE
      N = NQ
      C = XEND
      IRET = 3
      ARG = XSTART
      GO TO 300
  120 CONTINUE
      FX = RES
      IRET = 2
      ARG = XEND
      GO TO 300
  130 CONTINUE
      FD = RES
      FOLD = FD
C
      SUM1 = (FX+FD)/TWO
      SUM2 = (FX*FX+FD*FD)/TWO
C
      XMOLD = 1080.0
      XVOLD = 0.0
  140 CONTINUE
      NR = N - 1
      XOLD = XSTART
      H = (XEND-XSTART)/FLOAT(N)
      K = 1
      KD = 0
  150 CONTINUE
      IF (K .GT. NR) GO TO 180
C
      XOLD = XOLD + H
      IF (N.NE.NQ .AND. KD.EQ.1) GO TO 170
      IRET = 1
      ARG = XOLD
      GO TO 300
C
  160 CONTINUE
      FD = RES
      IF (FX*FD .LE. ZERO) GO TO 200
      SUM1 = SUM1 + FD
      SUM2 = SUM2 + FD*FD
  170 CONTINUE
      K = K + 1
      KD = KD + 1
      IF (KD .EQ. 2) KD = 0
      GO TO 150
  180 CONTINUE
```

```
      IF (FOLD*FX .LE. ZERO) GO TO 200
      AK = DFLOAT(K)
      XMEAN2 = (SUM1/AK)**2
      XVAR = SUM2/AK - XMEAN2
C
      IF (DABS(XMOLD*XVAR-XMEAN2*XVOLD) .LT. EPS1*DABS(XMEAN2*XVOLD))
     1    GO TO 190
C
      XMOLD = XMEAN2
      XVOLD = XVAR
      N = 2*N
      GO TO 140
  190 CONTINUE
      IF (XMEAN2 .GT. T2*XVAR) GO TO 220
      IF (I + 5 .GT. L) L = 0
      IF (L .EQ. 0) GO TO 260
C
      I = I + 2
      BOUNDS(I) = XSTART
      BOUNDS(I+1) = (XEND+XSTART)/TWO
      I = I + 2
      BOUNDS(I) = BOUNDS(I-1)
      BOUNDS(I+1) = XEND
      GO TO 220
C
  200 CONTINUE
      IF (I + 7 .GT. L) L = 0


      IF (L .EQ. 0) GO TO 260
      I = I + 2
      BOUNDS(I) = XSTART
C
      IF (K.NE.1 .AND. K.LT.N) XOLD = XOLD - H
      BOUNDS(I+1) = XOLD
      BOUNDS(I+2) = XOLD
      I = I + 2
      IF (K.EQ.1 .OR. K.GE.N) GO TO 210
      BOUNDS(I+1) = XOLD + H
      BOUNDS(I+2) = XOLD + H
      I = I + 2
  210 CONTINUE
      BOUNDS(I+1) = XEND
  220 CONTINUE
      IF (I .LE. 0) GO TO 260
C
      XSTART = BOUNDS(I)
      XEND = BOUNDS(I+1)
      I = I - 2
      XMEAN = DABS(XEND+XSTART)/TWO
C
      IF (XMEAN .LE. E) XMEAN = 1.0
      IF (XEND - XSTART .LT. E*XMEAN) GO TO 230
      GO TO 110
C
  230 CONTINUE
      IF (J + 2 .GT. L) L = 0
      IF (L .EQ. 0) GO TO 260
C
      J = J + 1
      WS(J) = XSTART
      IRET = 4
      ARG = XSTART
      GO TO 300
```

```
C
  240 CONTINUE
      WS1(J) = RES
      J = J + 1
      WS(J) = XEND
      IRET = 5
      ARG = XEND
      GO TO 300
  250 CONTINUE
      WS1(J) = RES
      GO TO 220
C
  260 CONTINUE
      IF (J .EQ. 0) GO TO 350
C
  270 CONTINUE
      XSTART = WS(J-1)
      IF (J .EQ. 2) GO TO 290
C
  280 CONTINUE
      XMEAN = DABS(WS(J-3))
      IF (XMEAN .LT. 1.0) XMEAN = 1.0


      IF (DABS(WS(J-3)-WS(J)) .GT. EPS2*XMEAN) GO TO 290
      IF (WS1(J)*WS1(J-1).LT.ZERO .AND. WS1(J-2)*WS1(J-3).LT.ZERO)
     1    GO TO 290
C
      J = J - 2
      IF (J .GE. 4) GO TO 280
C
  290 CONTINUE
      NROOT = NROOT + 1
      BOUNDS(NROOT) = XSTART
      NROOT = NROOT + 1
      BOUNDS(NROOT) = WS(J)
      J = J - 2
C
      IF (J .GE. 2) GO TO 270
      GO TO 350
  300 CONTINUE
      NRE = 0
      ARG1 = DABS(ARG)
      LLL = DIGITS - INT(DLOG10(ARG1+EPS3)+0.9)
C
      AHASH = ARG1*10.0**LLL + 0.5
      AHASH = DMOD(AHASH,ASIZE)
      NHASH = INT(AHASH) + 1
      ARG1 = DABS(ARG)
      IF (ARG1 .LE. 1.0) ARG1 = 1.0
  310 CONTINUE
      IF (DABS(STORE1(NHASH)-ARG) .LE. ARG1*EPS2) GO TO 330
      IF (STORE1(NHASH) .LE. C) GO TO 320
      STORE1(NHASH) = ARG
      RES = FVAL(ARG)
      STORE(NHASH) = RES
      GO TO (160,130,120,240,250) IRET
C
  320 CONTINUE
      NRE = NRE + 1
      IF (NRE .GT. LIMIT) GO TO 340
C
      NHASH = NHASH + NRE*NRE
      NHASH = MOD(NHASH,ISIZE) + 1
```

```
      GO TO 310
C
  330 CONTINUE
      RES = STORE(NHASH)
      GO TO (160,130,120,240,250) IRET
C
  340 CONTINUE
      RES = FVAL(ARG)
      GO TO (160,130,120,240,250) IRET
C
  350 CONTINUE
      NROOT = NROOT/2
      RETURN
      END
C
C ......................................................................
C
C       INCLUDE(PROCESS)
        FUNCTION FVAL(X)
        IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C***************************************************************
C
        COMMON /OTTO/ AC,BC,FK1,FK2,SIG1N,SIG2N,SIG3N,FCP,AAA,BBB,FT,IFG
      PI = 3.141592654
C
      COS3TH = 0.0
      FVAL = 0.0
      AI1 = 0.0
      AJ2 = 0.0
      AJ3 = 0.0
      IF (IFG .EQ. 1) THEN
          AI1 = SIG1N + SIG2N + X
          AJ2 = ((SIG1N-SIG2N)**2+(SIG2N-X)**2+(X-SIG1N)**2)/6.0
          AJ3 = (SIG1N-AI1/3.)*(SIG2N-AI1/3.)*(X-AI1/3.)
      ELSE IF (IFG .EQ. 2) THEN
          RATIO1 = SIG1N/SIG3N
          RATIO2 = SIG2N/SIG3N
          AI1 = RATIO1*X + RATIO2*X + X
          AJ2 = (((RATIO1-RATIO2)*X)**2+(RATIO2*X-X)**2+(X-RATIO1*X)**2)
     1        /6.0
          AJ3 = (RATIO1*X-AI1/3.)*(RATIO2*X-AI1/3.)*(X-AI1/3.)
      ENDIF
C    $            'RATIO1=',RATIO1,'RATIO2=',RATIO2
C
      IF (AJ2.GE.(-0.001) .AND. AJ2.LE.0.001) AJ2 = 0.001
C
      COS3TH = (3.0*DSQRT(3.0D00)/2.0)*(AJ3/DSQRT(AJ2*AJ2*AJ2))
C
      IF (COS3TH .GE. 0.0) THEN
          ALAMB = FK1*DCOS((1/3.)*DACOS(FK2*COS3TH))
      ELSE
          ALAMB = FK1*DCOS(PI/3.-(1/3.)*DACOS((-FK2*COS3TH)))
      ENDIF
C
      FVAL = AC*AJ2/(FCP*FCP) + ALAMB*DSQRT(AJ2)/FCP + BC*AI1/FCP - 1.0
```

```
C
      RETURN
      END

C**********************************************************
C     INCLUDE(PROCESS)
      SUBROUTINE FINDEX(FVALUE,FCP,PROPER,SIG1N,SIG2N,SIG3N,IOUT)
C
        IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C***********************************************************
        DIMENSION PROPER(25)
C
C THIS SUBROUTINE CHECKS TO SEE IF THE FAILURE ENVELOPE HAS BEEN
C VIOLATED IN THE CURRENT ITTERATION.
C FVALUE LE 0.0 NO FAILURE
C FAVLUE = 0.0 JUST FAILED
C FVALUE GE 0.0 VIOLATED ENVELOPE
C
      FVALUE = 0.0
      COS3TH = 0.0
      PI = 3.141592654
C       FCP=PROPER(5)
      AC = PROPER(18)
      BC = PROPER(19)
      FK1 = PROPER(20)
      FK2 = PROPER(21)
C

      AI1 = SIG1N + SIG2N + SIG3N
      AJ2 = ((SIG1N-SIG2N)**2+(SIG2N-SIG3N)**2+(SIG3N-SIG1N)**2)/6.0
C
      IF (AJ2.GE.(-0.001) .AND. AJ2.LE.0.001) AJ2 = 0.001
C
      AJ3 = (SIG1N-AI1/3.)*(SIG2N-AI1/3.)*(SIG3N-AI1/3.)
      COS3TH = (3.*DSQRT(3.0D00)/2.0)*(AJ3/AJ2**1.5)
C
      IF (COS3TH .GE. 0.0) THEN
          ALAMB = FK1*DCOS((1/3.)*DACOS(FK2*COS3TH))
      ELSE
          ALAMB = FK1*DCOS(PI/3.-(1/3.)*DACOS((-FK2*COS3TH)))
      ENDIF
C
      FVALUE=AC*AJ2/(FCP*FCP)+ALAMB*DSQRT(AJ2)/FCP+BC*AI1/FCP-1.0
C
      RETURN
      END
C***********************************************************
C
C     INCLUDE(PROCESS)
      SUBROUTINE PSTSTN(SSTEN,PVAL,PDIR,IFG,IOUT)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
```

```
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C DEVCSF
C
C*************************************************************
      DIMENSION SSTEN(6),PVAL(3),PDIR(3,3),A(3,3),EVAL(3),EVEC(3,3)
C
C     THIS SUBROUTINE DETERMINES THE PRINCIPAL STRESSES AND STRAINS.
C
C
C   IFG=1...STRAIN AND IFG=2... STRESS
C
      DO 100 IMN = 1, 3
          PDIR(IMN,1) = 0.0
          A(IMN,1) = 0.0
          EVEC(IMN,1) = 0.0
          PDIR(IMN,2) = 0.0
          A(IMN,2) = 0.0
          EVEC(IMN,2) = 0.0
          PDIR(IMN,3) = 0.0
          A(IMN,3) = 0.0
          EVEC(IMN,3) = 0.0
          PVAL(IMN) = 0.0
          EVAL(IMN) = 0.0
  100 CONTINUE
      PE1 = SSTEN(1)
      PE2 = SSTEN(2)
      PE3 = SSTEN(3)
      PE4 = SSTEN(4)
      PE5 = SSTEN(5)
      PE6 = SSTEN(6)
C
      A(1,1) = PE1
      A(2,2) = PE2
      A(3,3) = PE3
      IF (IFG .EQ. 1) THEN
          A(1,2) = PE4/2.0
          A(2,1) = PE4/2.0
          A(1,3) = PE6/2.0
          A(3,1) = PE6/2.0
          A(2,3) = PE5/2.0
          A(3,2) = PE5/2.0
      ELSE
          A(1,2) = PE4
          A(2,1) = PE4
          A(1,3) = PE6
          A(3,1) = PE6
          A(2,3) = PE5
          A(3,2) = PE5
      ENDIF
C
C  CALL IMSL ROUTINE FOR PRIN. VALUES.
C
      CALL DEVCSF (3, A, 3, EVAL, EVEC, 3)
C
      IM = 3
      P = EVEC(1,IM)
      Q = EVEC(2,IM)
      R = EVEC(3,IM)
      SQ = DSQRT(P*P+Q*Q+R*R)
      PDIR(1,1) = P/SQ
      PDIR(1,2) = Q/SQ
      PDIR(1,3) = R/SQ
```

```
      PVAL(1) = EVAL(IM)
      IM = 2
      P = EVEC(1,IM)
      Q = EVEC(2,IM)
      R = EVEC(3,IM)
      SQ = DSQRT(P*P+Q*Q+R*R)
      PDIR(2,1) = P/SQ
      PDIR(2,2) = Q/SQ
      PDIR(2,3) = R/SQ
      PVAL(2) = EVAL(IM)
      IM = 1
      P = EVEC(1,IM)
      Q = EVEC(2,IM)
      R = EVEC(3,IM)
      SQ = DSQRT(P*P+Q*Q+R*R)
      PDIR(3,1) = P/SQ
      PDIR(3,2) = Q/SQ
      PDIR(3,3) = R/SQ
      PVAL(3) = EVAL(IM)
C
      DO KM = 1, 3
          DO KN = 1, 3
              IF (DABS(PDIR(KM,KN)) .LE. 0.02) PDIR(KM,KN) = 0.0
          END DO
      END DO
      P = PDIR(1,1)
      Q = PDIR(1,2)
      R = PDIR(1,3)
      SQ = DSQRT(P*P+Q*Q+R*R)
      PDIR(1,1) = P/SQ
      PDIR(1,2) = Q/SQ
      PDIR(1,3) = R/SQ
      P = PDIR(2,1)
      Q = PDIR(2,2)
      R = PDIR(2,3)
      SQ = DSQRT(P*P+Q*Q+R*R)
      PDIR(2,1) = P/SQ
      PDIR(2,2) = Q/SQ
      PDIR(2,3) = R/SQ
      P = PDIR(3,1)
      Q = PDIR(3,2)
      R = PDIR(3,3)
      SQ = DSQRT(P*P+Q*Q+R*R)
      PDIR(3,1) = P/SQ
      PDIR(3,2) = Q/SQ
      PDIR(3,3) = R/SQ
C
      RETURN
      END
C
C
C _________________________ D  T  R  A  N  S _________________________
C
C     INCLUDE(PROCESS)
      SUBROUTINE DTRANS(D,PDIR,IOUT)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
```

```
C
C************************************************************
      DIMENSION D(6,*),T(6,6),DT(6,6),PDIR(3,3)
C
C
C              SUBROUTINE TO ROTATE THE MATERIAL CONSTITUTIVE MATRIX
C              TO THE GLOBAL AXES GIVEN THE DIRECTION COSINES. THE
C              ROTATED MATRIX WILL BE RETURNED IN ARRAY 'D'.
C
C
C
C               GENERATE THE TRANSFORMATION MATRIX 'T'
C
      AL1 = PDIR(1,1)
      AM1 = PDIR(1,2)
      AN1 = PDIR(1,3)
      AL2 = PDIR(2,1)
      AM2 = PDIR(2,2)
      AN2 = PDIR(2,3)
      AL3 = PDIR(3,1)
      AM3 = PDIR(3,2)
      AN3 = PDIR(3,3)
C
      IF (DABS(AL1) .LE. 0.01) AL1 = 0.0
      IF (DABS(AM1) .LE. 0.01) AM1 = 0.0
      IF (DABS(AN1) .LE. 0.01) AN1 = 0.0
      IF (DABS(AL2) .LE. 0.01) AL2 = 0.0
      IF (DABS(AM2) .LE. 0.01) AM2 = 0.0
      IF (DABS(AN2) .LE. 0.01) AN2 = 0.0
      IF (DABS(AL3) .LE. 0.01) AL3 = 0.0
      IF (DABS(AM3) .LE. 0.01) AM3 = 0.0
      IF (DABS(AN3) .LE. 0.01) AN3 = 0.0
C
      T(1,1) = AL1*AL1
      T(1,2) = AM1*AM1
      T(1,3) = AN1*AN1
      T(1,4) = AL1*AM1
      T(1,5) = AM1*AN1
      T(1,6) = AN1*AL1
      T(2,1) = AL2*AL2
      T(2,2) = AM2*AM2
      T(2,3) = AN2*AN2
      T(2,4) = AL2*AM2
      T(2,5) = AM2*AN2
      T(2,6) = AN2*AL2


      T(3,1) = AL3*AL3
      T(3,2) = AM3*AM3
      T(3,3) = AN3*AN3
      T(3,4) = AL3*AM3
      T(3,5) = AM3*AN3
      T(3,6) = AN3*AL3
      T(4,1) = 2.0*AL1*AL2
      T(4,2) = 2.0*AM1*AM2
      T(4,3) = 2.0*AN1*AN2
      T(4,4) = AL1*AM2 + AL2*AM1
      T(4,5) = AM1*AN2 + AM2*AN1
      T(4,6) = AN1*AL2 + AN2*AL1
      T(5,1) = 2.0*AL2*AL3
      T(5,2) = 2.0*AM2*AM3
      T(5,3) = 2.0*AN2*AN3
      T(5,4) = AL3*AM2 + AL2*AM3
      T(5,5) = AM3*AN2 + AM2*AN3
```

```
      T(5,6) = AN3*AL2 + AN2*AL3
      T(6,1) = 2.0*AL1*AL3
      T(6,2) = 2.0*AM1*AM3
      T(6,3) = 2.0*AN1*AN3
      T(6,4) = AL1*AM3 + AL3*AM1
      T(6,5) = AM1*AN3 + AM3*AN1
      T(6,6) = AN1*AL3 + AN3*AL1
C
C               D(GLOBAL)=T(TRANSPOSE)*D(MATERIAL AXES)*T
C
C          .... PERFORM D*T=DT
C
      DO 120 I = 1, 6
          DO 110 J = 1, 6
              DT(I,J) = 0.0
              DO 100 K = 1, 6
                  DT(I,J) = DT(I,J) + D(I,K)*T(K,J)
  100         CONTINUE
  110     CONTINUE
  120 CONTINUE
C
C
C               PERFORM D(XY)=T TRANS*DT
C
      DO 150 I = 1, 6
          DO 140 J = 1, 6
              D(I,J) = 0.0
              DO 130 K = 1, 6
                  D(I,J) = D(I,J) + T(K,I)*DT(K,J)
  130         CONTINUE
  140     CONTINUE
  150 CONTINUE
C
      RETURN
      END
C ----------------------------  C  H  K  S  T  L  ---------------------
C
C     INCLUDE(PROCESS)
      SUBROUTINE CHKSTL(PROPER,TOLER,NSTIFF,CONVER,IELEM,L,ITYPE
     $     ,IOUT)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C IOGET


C
C*************************************************************
      CHARACTER*105,DUMMY
      REAL*8 LTHICK,LENGTH
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON /BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $,IHIST2
      COMMON /ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
      COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
      COMMON/UTIL1/STRESS(6),STRAIN(6),DUMMY
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      DIMENSION PROPER(25)
      LOGICAL CHKGUS
C
```

```
C
C      SUBROUTINE TO CHECK CONVERGENCE AT EACH GAUSS POINT
C      OF STEEL LAYER 'J' OF ELEMENT 'I'.IF CONVERGENCE IS
C      NOT ACHIEVED THE CONSTITUTIVE MATRIX FOR THE POINT
C      AND THE STIFFNESS MATRIX FOR THE ELEMENT WILL BE
C      REGENERATED AND MORE ITERATIONS WILL BE CARRIED OUT.
C
C
C
C               DETERMINE THE STRAIN AT YIELD.
C
      EPSNEW = 0.0
      ESNEW = 0.0
      J = LYNUM
      CALL IOGET (LDEV1, 96, '(A96)', 5)
      EPSX = STRAIN(1)
      EPSY = STRAIN(2)
      EPSZ = STRAIN(3)
      GAMAXY = STRAIN(4)
      GAMAYZ = STRAIN(5)
      GAMAXZ = STRAIN(6)
C
C FOR EACH INTEGRATION POINT OF EACH LAYER AND ELEMENT RECOVER THE
C STRAINS AND PROCEED WITH THE CONVERGENCE CHECK
C
      SIGYLD = HISTO(L,13,J)
      ESYLD = HISTO(L,14,J)
      EPSYLD = HISTO(L,12,J)
C
C               REDUCE TRUNCATION ERRORS ...
C
      XYZ = DMAX1(DMAX1(DMAX1(DMAX1(DMAX1(DABS(EPSX),DABS(EPSY)),DABS(
     1    GAMAXY)),DABS(GAMAYZ)),DABS(GAMAXZ)),DABS(EPSZ))
      IF (DABS(EPSX) .LT. XYZ*10.0E-7) EPSX = 0.0
      IF (DABS(EPSY) .LT. XYZ*10.0E-7) EPSY = 0.0
      IF (DABS(EPSZ) .LT. XYZ*10.0E-7) EPSZ = 0.0
      IF (DABS(GAMAXY) .LT. XYZ*10.0E-7) GAMAXY = 0.0
      IF (DABS(GAMAYZ) .LT. XYZ*10.0E-7) GAMAYZ = 0.0
      IF (DABS(GAMAXZ) .LT. XYZ*10.0E-7) GAMAXZ = 0.0
C
C               CALCULATE THE CURRENT STRAIN IN THE DIRECTION OF REBARS
C
      AL1 = HISTO(L,17,J)
      AM1 = HISTO(L,18,J)
      AN1 = HISTO(L,19,J)
      EPSSTL = EPSX*AL1*AL1 + EPSY*AM1*AM1 + EPSZ*AN1*AN1 + GAMAXY*AL1*
     1    AM1 + GAMAYZ*AM1*AN1 + GAMAXZ*AN1*AM1
      IF (DABS(EPSSTL) .LE. 1E-7) EPSSTL = 0.0
C
C               COMPARE THIS WITH THE PREVIOUS STRAIN.
C
      EPSOLD = HISTO(L,10,J)
      IF (DABS(EPSSTL) .GE. .999*DABS(EPSOLD)) THEN
C
C               CHANGE THE STATE OF THE POINT TO LOADING
C
          IF (HISTO(L,16,J).EQ.2. .OR. HISTO(L,16,J).EQ.3.) HISTO(L,16,J
     1        ) = 1.0
          IF (HISTO(L,16,J) .EQ. 1.) THEN
C
C
C                             -----------
C                             * LOADING *
```

```
C                         -----------
C
C
C
C              CONSIDER POSSIBILITY OF UNLOADING IN THE NEXT STEPS.
C              THEREFORE,SAVE THE STRAIN FROM WHICH UNLOADING MIGHT
C              INITIATE.
C
C
            HISTO(L,10,J) = EPSSTL
C
C              SAVE THE CURRENT STRAIN.
C
            HISTO(L,11,J) = EPSSTL
C
C              COMPARE THE CURRENT STRAIN WITH THE YIELD STRAIN.
C
            IF (DABS(EPSSTL) .LE. DABS(EPSYLD)) RETURN
C
C
C             _________ POST-YIELD CALCULATIONS _________
C
            HISTO(L,15,J) = 2.
C
C              DETERMINE SECANT YOUNG'S MODULUS CORRESPONDING TO EPSST
C
C              ALFA = HARDENING PARAMETER
C
            ALFA = PROPER(6)
C     ALFA=0.01
            EINIT = HISTO(L,14,J)
            EPSULT = PROPER(7)
            SIGULT = SIGYLD*(1.0-ALFA) + ALFA*EINIT*DABS(EPSULT)
C
C       TOTAL AREA UNDER STRESS-STRAIN CURVE
C
            AREA = 0.5*SIGYLD*DABS(EPSYLD) + (DABS(EPSULT)-DABS(EPSYLD
     1          ))*SIGYLD + (SIGULT-SIGYLD)*(DABS(EPSULT)-DABS(EPSYLD)
     2          )*0.50
C
C     FIND THE STIFFNESS CORRESPONDING TO THE UPDATED STRAIN
C
            ES = (SIGYLD*(1.-ALFA)+ALFA*EINIT*DABS(EPSSTL))/DABS(
     1          EPSSTL)
C
            SUBARA = 0.5*(HISTO(L,1,J)*DABS(EPSOLD)+ES*DABS(EPSSTL))*(
     1          DABS(EPSSTL)-DABS(EPSOLD)) - 0.5*DABS(EPSSTL)*DABS(
     2          EPSSTL)*ES + 0.5*DABS(EPSOLD)*DABS(EPSOLD)*HISTO(L,1,J
     3          )
            IFLAG = 1
            PERCNT = (SUBARA/AREA)*100
            IF (PERCNT .GT. TOLER) THEN
                ESNEW = ES
                IFLAG = 0
            ENDIF
            TOLERE = 1.0
            SIGCUR = HISTO(L,1,J)*DABS(EPSSTL)
            IF (SIGCUR.GT.SIGULT .AND. IFLAG.EQ.1) THEN
                ARA = 0.5*SIGYLD*DABS(EPSYLD) + (DABS(EPSOLD)-DABS(
     1              EPSYLD))*SIGYLD + (HISTO(L,1,J)*DABS(EPSOLD)-
     2              SIGYLD)*(DABS(EPSOLD)-DABS(EPSYLD))*0.5 + AREA*
     3              TOLERE/100.0 - 0.5*DABS(EPSOLD)*DABS(EPSOLD)*HISTO
     4              (L,1,J)
```

```
C
C... LENGTH CORRECT FOR SMALL ALFA'S
C
                 LENGTH = DSQRT(SIGYLD*SIGYLD+EPSYLD*EPSYLD)
C
                 ALTUDE = ARA/(LENGTH*0.5)
C
                 EPSNEW = ALTUDE + EPSYLD
                 ESNEW = (SIGYLD*(1-ALFA)+ALFA*EINIT*DABS(EPSNEW))/
     1               DABS(EPSNEW)
                 IFLAG = 0
                 HISTO(L,10,J) = EPSNEW
                 HISTO(L,11,J) = EPSNEW
             ENDIF
             IF (PERCNT.GT.TOLER .OR. IFLAG.EQ.0) THEN
C
C               IN ORDER TO UPDATE THE CONSTITUTIVE LAW AT THE POINT
C               SIMPLY DIVIDE THE PREVIOUSLY ROTATED CONSTITUTIVE
C               MATRIX STORED IN ARRAY 'CRACK' BY THE OLD 'E' AND
C               THEN MULTIPLY IT BY THE NEW SECANT 'E'.
C
C
                 RATIO = ESNEW/HISTO(L,1,J)
                 DO 100 K = 2, 37
                     CRACK(L,K,J) = CRACK(L,K,J)*RATIO
  100            CONTINUE
C
C               STORE THE CURRENT SECANT YOUNG'S MODULUS.
C
                 HISTO(L,1,J) = ESNEW
C
C
C               UPDATE ELEMENT STIFFNESS MATRIX AND CARRY OUT FURTHER
C               ITERATIONS.
C
C
                 NSTIFF = 1
                 CONVER = 999.
                 CHKGUS = .TRUE.
             ENDIF
         ENDIF
         RETURN
      ENDIF
C
C
C                          -------------
C                          * UNLOADING *
C                          -------------
C
C
      IF (DABS(EPSSTL) .LT. .999*DABS(EPSOLD)) THEN
C
C               GET THE STRAIN IN PREVIOUS ITERATIONS.
C
         PREEPS = HISTO(L,11,J)
         VAR = EPSSTL*PREEPS
         IF (VAR .LT. 0.0) THEN
             HISTO(L,10,J) = 0.0
             HISTO(L,11,J) = 0.0
             HISTO(L,16,J) = 1.0
             CONVER = 999.0
             RETURN
         ENDIF
```

```
          IF (DABS(EPSSTL) .LT. .999*DABS(PREEPS)) THEN
C
C                SET THE STATE OF THE POINT TO UNLOADING
C
             HISTO(L,16,J) = 2.
C
C                SAVE THE CURRENT STRAIN
C
             HISTO(L,11,J) = EPSSTL
             RETURN
          ENDIF
C
C
C                             -------------
C                             * RELOADING *
C                             -------------
C
C
          IF (DABS(EPSSTL) .GE. .999*DABS(PREEPS)) THEN
             HISTO(L,16,J) = 3.
C
C                SAVE THE CURRENT STRAIN
C
             HISTO(L,11,J) = EPSSTL
          ENDIF
      ENDIF
C
      RETURN
      END
C
C


C
C ----------------------------- S  I  M  U  L  -----------------------
C
C     INCLUDE(PROCESS)
      FUNCTION SIMUL(N,A,EPS,NRC,IOUT)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C**************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C**************************************************************
      DIMENSION IROW(40),JCOL(40),JORD(40),Y(40),A(NRC,*)
C     REAL*12   A,AIJCK,PIVOT,EPS
C
C
C               FUNCTION TO COMPUTE THE INVERSE OF THE N BY N
C               MATRIX A.THE VALUE OF THE DETERMINANT IS RET-
C               URNED AS THE VALUE OF THE FUNCTION.SHOULD THE
C               POTENTIAL PIVOT OF LARGEST MAGNITUDE BE SMALLER
C               IN MAGNITUDE THAN EPS,THE MATRIX IS CONSIDERED
C               TO BE SINGULAR AND A TRUE ZERO IS RETURNED AS
C               THE VALUE OF THE FUNCTION.
C
C
C          ..... N = NO. OF ROWS IN MATRIX A
C
      MAX = N
```

```
      DO 100 M = 1, 40
          IROW(M) = 0
          JCOL(M) = 0
          JORD(M) = 0
          Y(M) = 0.0
  100 CONTINUE
C
C               ELIMINATE ANY SMALL ELEMENT WHICH CAN CAUSE
C               OVERFLOW PROBLEM DURING MATRIX INVERSION.
C
C     ........  SEARCH FOR THE LARGEST ELEMENT
C
      BIG = 0.0
      DO 120 I = 1, N
          DO 110 J = 1, N
              BIG = DMAX1(DABS(A(I,J)),BIG)
  110     CONTINUE
  120 CONTINUE
C
      VERYSN = BIG*10.**(-10)
      DO 140 I = 1, N
          DO 130 J = 1, N
              IF (DABS(A(I,J)) .LE. VERYSN) A(I,J) = 0.0
  130     CONTINUE
  140 CONTINUE
C
C            ... IS N LARGER THAN 40
C
      IF (N .LE. 40) GO TO 150
      WRITE (*, 900) N
      SIMUL = 0.0
      RETURN
C
C               BEGIN ELIMINATION PROCEDURE
C
  150 CONTINUE
      DETER = 1.
      DO 270 K = 1, N
          KM1 = K - 1
C
C               SEARCH FOR THE PIVOT ELEMENT
C
          PIVOT = 0.0
          DO 210 I = 1, N
              DO 200 J = 1, N
C
C               SCAN IROW AND JCOL ARRAYS FOR INVALID PIVOT SUBSCRIPTS
C
                IF (K .EQ. 1) GO TO 180
                DO 170 ISCAN = 1, KM1
                    DO 160 JSCAN = 1, KM1
                        IF (I .EQ. IROW(ISCAN)) GO TO 190
                        IF (J .EQ. JCOL(JSCAN)) GO TO 190
  160               CONTINUE
  170           CONTINUE
  180           CONTINUE
                IF (DABS(A(I,J)) .LE. DABS(PIVOT)) GO TO 190
                PIVOT = A(I,J)
                IROW(K) = I
                JCOL(K) = J
  190           CONTINUE
  200         CONTINUE
  210     CONTINUE
```

```
C
C               INSURE THAT SELECTED PIVOT IS LARGER THAN EPS
C
          IF (DABS(PIVOT) .GT. EPS) GO TO 220
          SIMUL = 0.0
          RETURN
C
C               UPDATE THE DETERMINANT VALUE
C
  220     CONTINUE
          IROWK = IROW(K)
          JCOLK = JCOL(K)
          DETER = DETER*PIVOT
C
C               NORMALIZE PIVOT ROW ELEMENTS
C
          DO 230 J = 1, MAX
              A(IROWK,J) = A(IROWK,J)/PIVOT
  230     CONTINUE
C
C               CARRY OUT ELIMINATION AND DEVELOPE INVERSE
C
          A(IROWK,JCOLK) = 1./PIVOT
          DO 260 I = 1, N
              AIJCK = A(I,JCOLK)
              IF (I .EQ. IROWK) GO TO 250
              A(I,JCOLK) = -AIJCK/PIVOT
              DO 240 J = 1, MAX
                  IF (J .NE. JCOLK) A(I,J) = A(I,J) - AIJCK*A(IROWK,J)
  240         CONTINUE
  250         CONTINUE
  260     CONTINUE
  270 CONTINUE
C
C               ORDER SOLUTION VALUES (IF ANY) AND CREATE JORD ARRAY
C
      DO 280 I = 1, N
          IROWI = IROW(I)
          JCOLI = JCOL(I)
          JORD(IROWI) = JCOLI
  280 CONTINUE
C
C               ADJUST SIGN OF DETERMINANT
C
      INTCH = 0
      NM1 = N - 1
      DO 310 I = 1, NM1
          IP1 = I + 1
          DO 300 J = IP1, N
              IF (JORD(J) .GE. JORD(I)) GO TO 290
              JTEMP = JORD(J)
              JORD(J) = JORD(I)
              JORD(I) = JTEMP
              INTCH = INTCH + 1
  290         CONTINUE
  300     CONTINUE
  310 CONTINUE
      IF (INTCH/2*2 .NE. INTCH) DETER = -DETER
C
C               IF INDIC IS NEGATIVE OR ZERO, UNSCRAMBLE THE INVERSE
C          .... FIRST BY ROWS
C
```

```
  320 CONTINUE
      DO 350 J = 1, N
          DO 330 I = 1, N
              IROWI = IROW(I)
              JCOLI = JCOL(I)
              Y(JCOLI) = A(IROWI,J)
  330     CONTINUE
          DO 340 I = 1, N
              A(I,J) = Y(I)
  340     CONTINUE

  350 CONTINUE
C
C        ...... THEN BY COLUMNS
C
      DO 380 I = 1, N
          DO 360 J = 1, N
              IROWJ = IROW(J)
              JCOLJ = JCOL(J)
              Y(IROWJ) = A(I,JCOLJ)
  360     CONTINUE
          DO 370 J = 1, N
              A(I,J) = Y(J)
  370     CONTINUE
  380 CONTINUE
C
C               RETURN FOR INDIC NEGATIVE OR ZERO
C
      SIMUL = DETER
      RETURN
C
C               FORMAT FOR OUTPUT STATEMENT
C
  900 FORMAT(/,1X,'   ERROR IN SUBROUTINE SIMUL',/,5X,'N=',I3,
     1       1X,'WHICH EXCEEDS ITS MAX. VALUE OF 40')
      END
C
C ======================== S O L V E 1 ===========================
C
C     INCLUDE (PROCESS)
      SUBROUTINE SOLVE1(A,C,B,JDIAG,NEQ,AFAC,BACK)
C
C ==================================================================
C I
C I   P R O G R A M:
C I
C I   PROGRAM 'SOLVE1' IS USED TO SOLVE A SERIES OF BANDED
C I   NONSYMETRIC LINEAR EQUATIONS USING THE GAUSS ELIMINATION/BACK
C I   SUBSTITUTION WITH NO COLUMN PIVOTING.
C I
C I   STORAGE:    COEFICIANT MATRIX SHOULD BE STORED IN TWO ONE
C I               DIMENSIONAL ARRAYS USING THE SKYLINE OR THE
C I               PROFILE METHOD
C I               A( K ) = UPPER TRIANGULAR MATRIX
C I               C( K ) = LOWER TRIANGULAR MATRIC
C I               B( K ) = RIGHT HAND SIDE VECTOR ON CALL
C I                      = VECTOR OF UNKNOWNS ON RETURN
C I
C ==================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
```

```
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C DOTPRO
C
C***********************************************************
      LOGICAL AFAC,BACK
      DIMENSION A( 1 ),C( 1 ),B( 1 ),JDIAG( 1 )
C
C         FACTOR A TO UT*D*U, REDUCE B TO Y
C
      JR = 0
      DO 140 J = 1, NEQ
          JD = JDIAG(J)
          JH = JD - JR
          IF (JH .LE. 1) GO TO 130
          IS = J + 1 - JH
          IE = J - 1
          IF (.NOT.AFAC) GO TO 120
          K = JR + 1
          ID = 0
C
C        REDUCE ALL EQUATIONS EXCEPT DIAGNAL
C
          DO 110 I = IS, IE
              IR = ID
              ID = JDIAG(I)
              IH = MINO(ID-IR-1,I-IS)
              IF (IH .EQ. 0) GO TO 100
              A(K) = A(K) - DOTPRO(A(K-IH),C(ID-IH),IH)
              C(K) = C(K) - DOTPRO(C(K-IH),A(ID-IH),IH)
 100          CONTINUE
              IF (A(ID) .NE. 0.0) C(K) = C(K)/A(ID)
              K = K + 1
 110      CONTINUE
C
C         REDUCE THE DIAGNAL TERM
C
          A(JD) = A(JD) - DOTPRO(A(JR+1),C(JR+1),JH-1)
C
C         FORWARD REDUCE THE RIGHT HAND SIDE
C
 120      CONTINUE
          IF (BACK) B(J) = B(J) - DOTPRO(C(JR+1),B(IS),JH-1)
 130      CONTINUE
          JR = JD
 140  CONTINUE
      IF (.NOT.BACK) RETURN
C
C         BACK SUBSTITUTION
C
      J = NEQ
      JD = JDIAG(J)
 150  CONTINUE
      IF (A(JD) .NE. 0.0) B(J) = B(J)/A(JD)
      D = B(J)
      J = J - 1
      IF (J .LE. 0) RETURN
      JR = JDIAG(J)
      IF (JD - JR .LE. 1) GO TO 170
      IS = J - JD + JR + 2
      K = JR - IS + 1
```

```
      DO 160 I = IS, J
          B(I) = B(I) - A(I+K)*D
  160 CONTINUE
  170 CONTINUE
      JD = JR
      GO TO 150
      END


C
C ======================== S O L V E 2 ===========================
C
C     INCLUDE (PROCESS)
      SUBROUTINE SOLVE2(A,R,JDIAG,NEQU,KKK,IOUT)
C
C ===============================================================
C I
C I   THIS PROGRAM IS USED TO SOLVE FINITE ELEMENT STATIC EQUILIB.
C I   EQUATIONS IN CORE, USING COMPACTED STORAGE AND COLUMN REDUCTON
C I   SCHEME
C I
C ===============================================================
C
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C***********************************************************
      DIMENSION A( 1 ), JDIAG( 1 ), R( 1 )
C
C        PERFORM L*D*L FACTORIZATION OF THE STIFFNESS MATRIX
C
      IF (KKK .LT. 2) THEN
          DO 140 N = 1, NEQU
              KN = JDIAG(N)
              KL = KN + 1
              KU = JDIAG(N+1) - 1
              KH = KU - KL
              IF (KH .GT. 0) THEN
                  K = N - KH
                  IC = 0
                  KLT = KU
                  DO 110 J = 1, KH
                      IC = IC + 1
                      KLT = KLT - 1
                      KI = JDIAG(K)
                      ND = JDIAG(K+1) - KI - 1
                      IF (ND .GT. 0) THEN
                          KK = MINO(IC,ND)
                          C = 0.
                          DO 100 L = 1, KK
                              C = C + A(KI+L)*A(KLT+L)
  100                     CONTINUE
                          A(KLT) = A(KLT) - C
                      ENDIF
                      K = K + 1
  110             CONTINUE
              ELSE IF (KH .LT. 0) THEN
                  GO TO 130
```

```
               ENDIF
C
               K = N
               B = 0.
C*VDIR: IGNORE RECRDEPS
               DO 120 KK = KL, KU
                   K = K - 1
                   KI = JDIAG(K)
                   C = A(KK)/A(KI)
                   B = B + C*A(KK)
                   A(KK) = C
  120          CONTINUE
               A(KN) = A(KN) - B
  130          CONTINUE
               IF (DABS(A(KN)) .LE. 1.E-9) THEN
                   WRITE (IOUT, 900) N, A(KN)
                   STOP
               ENDIF
  140      CONTINUE
           RETURN
C
C        REDUCE THE RIGHT-HAND-SIDE LOAD VECTOR
C
       ELSE
           DO 160 N = 1, NEQU
               KL = JDIAG(N) + 1
               KU = JDIAG(N+1) - 1
               IF (KU - KL .GE. 0) THEN
                   K = N
                   C = 0.
                   DO 150 KK = KL, KU
                       K = K - 1
                       C = C + A(KK)*R(K)
  150              CONTINUE
                   R(N) = R(N) - C
               ENDIF
  160      CONTINUE
C
C        BACK-SUBSTITUTE
C
           DO 170 N = 1, NEQU
               K = JDIAG(N)
               R(N) = R(N)/A(K)
  170      CONTINUE
C
           IF (NEQU .EQ. 1) RETURN
           N = NEQU
           DO 190 L = 2, NEQU
               KL = JDIAG(N) + 1
               KU = JDIAG(N+1) - 1
               IF (KU - KL .GE. 0) THEN
                   K = N
C*VDIR: IGNORE RECRDEPS
                   DO 180 KK = KL, KU
                       K = K - 1
                       R(K) = R(K) - A(KK)*R(N)
  180              CONTINUE
               ENDIF
               N = N - 1
  190      CONTINUE
       ENDIF
       RETURN
  900 FORMAT(//1X,'STOP - STIFFNESS MATRIX NOT POSITIVE DEFINITE '//
```

```
     1 1X,' NONPOSITIVE PIVOT FOR EQUATION ',I4//1X,'PIVOT = ',E20.12)
      END

C
C ============================ D O T ===============================
C
C     INCLUDE (PROCESS)
      FUNCTION DOTPRO(A,B,N)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C***************************************************************
      REAL*8 A,B,DOTPRO,TEMP
      DIMENSION A( 1 ) , B( 1 )
      TEMP = 0.0
      DO 100 I = 1, N
          TEMP = TEMP + A(I)*B(I)
  100 CONTINUE
      DOTPRO = TEMP
C
      RETURN
      END
C
C ========================== C R O S S ============================
C
      SUBROUTINE CROSS(V1,V2,V3)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C***************************************************************
      DIMENSION V1(3),V2(3),V3(3)
      V3(1) = V1(2)*V2(3) - V1(3)*V2(2)
      V3(2) = V1(3)*V2(1) - V1(1)*V2(3)
      V3(3) = V1(1)*V2(2) - V1(2)*V2(1)
      RETURN
      END
C
C ========================== U N I T V ============================
C
      SUBROUTINE UNITV(V1,N)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C***************************************************************
      DIMENSION V1(1)
```

```
C
      SQSUM = 0.DO
      DO 100 K1 = 1, N
          SQSUM = SQSUM + V1(K1)**2
  100 CONTINUE
      VNORM = DSQRT(SQSUM)
C
      IF (VNORM .NE. 0.0DO) THEN
          DO 110 K1 = 1, N
              V1(K1) = V1(K1)/VNORM
  110     CONTINUE
      ENDIF
C
      RETURN
      END
C
C ============================ U N I T S ==============================
C
      SUBROUTINE UNITVS(V1,N)


      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C**************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C**************************************************************
      REAL*4 V1
      DIMENSION V1(1)
C
      SQSUM = 0.DO
      DO 100 K1 = 1, N
          SQSUM = SQSUM + V1(K1)**2
  100 CONTINUE
      VNORM = DSQRT(SQSUM)
C
      IF (VNORM .NE. 0.0DO) THEN
          DO 110 K1 = 1, N
              V1(K1) = V1(K1)/VNORM
  110     CONTINUE
      ENDIF
C
      RETURN
      END
C
C ====================== D I R V E C ===============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE DIRVEC(V1,V2,V3)
C
C ====================================================================
C I
C I   P R O G R A M:
C I
C I   DIRCOS EVALUATES THE DIRECTION COSINES OF THE Y_PRIME AND
C I   THE X_PRIME AXES GIVEN THE Z_PRIME DIRECTION.
C I
C I
C I   A R G U M E N T    L I S T
C I
```

```
C I   V1      = vector containing the direction cosines of the x-prime
C I   V2      = vector containing the direction cosines of the y-prime
C I   V3      = vector containing the direction cosines of the z-prime
C I
C ====================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C************************************************************
      DIMENSION V1( 3 ),V2( 3 ),V3( 3 )
C
C
C ----- THIRD UNIT VECTOR IS USER SPECIFIED THE OTHER TWO ARE EVALUATED
C         AS DISCRIBED BY COMMENT LINES
C
      TX = V3(1)
      TY = V3(2)
      TZ = V3(3)
C
C ----- THE FIRST VECTOR IS CHOSEN TO BE (V3 X K)
C
      CNORM = DSQRT(TX**2+TY**2)
      IF (DABS(CNORM) .LT. 1.0D-8) THEN
          V1(1) = 1.
          V1(2) = 0.
          V1(3) = 0.
      ELSE
          V1(1) = TY/CNORM
          V1(2) = -TX/CNORM


          V1(3) = 0.
      ENDIF
C
C ----- THE SECOND VECTOR IS DEFINED AS (V3 X V1)
C
      VX = TY*V1(3) - TZ*V1(2)
      VY = (-TX*V1(3)) + TZ*V1(1)
      VZ = TX*V1(2) - TY*V1(1)
      CNORM = DSQRT(VX**2+VY**2+VZ**2)
      V2(1) = VX/CNORM
      V2(2) = VY/CNORM
      V2(3) = VZ/CNORM
C
      RETURN
      END
C====================================================================
C
C      INCLUDE(PROCESS)
       SUBROUTINE OUT28(IOUT)
       IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
```

```
C ELINFO    ELINTM    LYINFO    ISH3DG    ISHSHL
C ISH2DG    SHNORM    DIRVEC    JACB3D    COORD1    GETTHK
C JACSHL
C
C*************************************************************
       CHARACTER*80 ID1
       REAL*4 THICK,SHELLZ
       REAL*8 LTHICK,N,NXI,NETA,NSI
       INTEGER ELNUM,ELNUMB
       COMMON/OUTPT1/IOFLAG,IFCAED
       COMMON/BLOCK5/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
     $ ,IHIST2
       COMMON/SHLDIR/VECT(3,3,9)
       COMMON/ABC/TENSTF(27,80,15),ITNSPG,ITNSG1,ITNSG2
       COMMON/LPROP/PROPER(25,15)
       COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
       COMMON/INPUTF/MATYPE(10)
       COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
       COMMON/ISHAP1/N(20,27),NXI(20,27),NETA(20,27),NSI(20,27),SI(27)
       COMMON/ISHAP2/W(27)
       COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM
     $ ,NINODE,NCOLOR,NFREE
       COMMON/INCR11/FRACT(10),NLINC1(10),LDCONT,INCPTR
       COMMON/MATER1/DEP(6,6)
       COMMON/OUTPT2/IOEL(5000),IONOD(10000)
       COMMON/CONTR1/INCREM,NIT
       COMMON/INPUT2/NOP(20,5000)
       COMMON/INPUTE/ISPB(10000)
       COMMON/SKTR2/SHELLZ(3,10000)
       COMMON/INPUT9/THICK(9),IFLAG
       COMMON/ICKPIF/ICKFGO,ICKNTR
       DIMENSION PDIR(3,3),V3(3)
C
C      OUTPUT MATERIAL INFORMATION AT SAMPLE POINT OF EACH CONCRETE
C      /STEEL SAMPLE POINT. IT IS SET UP TO POST PROCESS CRACK INFO
C      FOR GRAPHICS PROCESSING ON CAEDS.
C
      ICRK = 40
      ILINE = 1
      ICKNTR = 10
      ICK2 = 72
      INTGFG = 0
      IF (BTEST(IOFLAG,1) .AND. BTEST(IOFLAG,2)) INTGFG = 1
C
      WRITE (ICRK, 1330) INCREM
      WRITE (ICRK, 1340) NIT
      REWIND IHISTY
      REWIND ITNSPG
      REWIND ITNSG1
      REWIND IHIST1
      IF (ICKFGO .EQ. 1) THEN
          IF (MOD(INCREM,10) .EQ. 0) THEN
              ID1 = 'CRACK_ORIENTATION'
              WRITE (ICK2, '(A6)') '    -1'
              WRITE (ICK2, '(A6)') '   800'
              WRITE (ICK2, 1430) INCREM
              WRITE (ICK2, 1440) ID1
              WRITE (ICK2, '(A6)') '    -1'
              WRITE (ICK2, '(A6)') '    -1'
              WRITE (ICK2, '(A6)') '   801'
              WRITE (*, *) 'ICKNTR', ICKNTR, 'INCREM', INCREM
          ENDIF
      ENDIF
```

```
      DO 250 ELNUM = 1, NELEM
          CALL ELINFO (ELNUM, ITYPE, NNEL, IFLAG, ISTART, LINES)
          CALL ELINTM (ELNUM, IDENT, INTCOD, NIPXI, NIPETA, NIPSI,
     1         MATNUM, THICK)
          DO 230 LYNUM = 1, NLAYRS
C
              D11 = 0.0
              D12 = 0.0
              D13 = 0.0
              D21 = 0.0
              D22 = 0.0
              D23 = 0.0
              D31 = 0.0
              D32 = 0.0
              D33 = 0.0
              PERCNT = 0.0
C
              CALL LYINFO (LYNUM, LTHICK, ZS, MATRL, DCS, NNEL, ELNUM,
     1            NIPXI, NIPETA, NIPSI)
C
C
              IF (MATYPE(MATNUM) .NE. 4) GO TO 240
              NIP = NIPXI*NIPETA*NIPSI
C     WRITE(*,*)'**********PROCESSING CONCRETE OUTPUT**********'
              MM = (ELNUM-1)*27 + 1
              READ (IHISTY) ELNUMB, LYNUMB, NGAUS, ((CRACK(L,K,LYNUM),K
     1             = 1,250), L = 1, NIP)
              IF (ELNUMB.NE.ELNUM .OR. LYNUMB.NE.LYNUM .OR. NGAUS.NE.NIP


     1            ) THEN
                  WRITE (*, *) 'ERROR IN READING IHISTY IN OUT28'
                  WRITE (*, *) 'ELNUM', ELNUM, 'ELNUMB', ELNUMB, 'LYNUM'
     1                , LYNUM, 'LYNUMB', LYNUMB, 'NGAUS', NGAUS, 'NIP',
     2                NIP
                  STOP
              ENDIF
              READ (ITNSPG) ELNUMB, LYNUMB, NGAUS, MATRL, ((TENSTF(L,K,
     1            LYNUM),K = 1,80), L = 1, NIP), ((HISTO(L,K,LYNUM),K =
     2            1,70), L = 1, NIP), (PROPER(KM,MATRL), KM = 1, 25)
              IF (ELNUMB.NE.ELNUM .OR. LYNUMB.NE.LYNUM .OR. NGAUS.NE.NIP
     1            ) THEN
                  WRITE (*, *) 'ERROR IN READING ITNSPG IN OUT28'
                  WRITE (*, *) 'ELNUM', ELNUM, 'ELNUMB', ELNUMB, 'LYNUM'
     1                , LYNUM, 'LYNUMB', LYNUMB, 'NGAUS', NGAUS, 'NIP',
     2                NIP
                  STOP
              ENDIF
C
C
C.... TYPE OF MATREIAL 0=STEEL
C                     1=CONCRETE
C                     2=ELASTIC
C
              IF (ITYPE .LE. 0) GO TO 240
              IF (ITYPE.NE.0 .OR. IDENT.NE.0) THEN
                  IF (ITYPE .GT. 300) THEN
                      IF (IFLAG .EQ. 0) THEN
                          NCB = 3*NNEL
                              CALL ISH3DG (ITYPE, NNEL, IERROR)
                      ELSE IF (IFLAG .EQ. 4) THEN
                          NCB = 6*NNEL
                          CALL ISHSHL (ITYPE, NNEL, IERROR)
                      ENDIF
```

```
                ELSE IF (ITYPE .GT. 200) THEN
                    NCB = 2*NNEL
                    CALL ISH2DG (ITYPE, NNEL, IERROR)
                    IF (IFLAG .EQ. 3) THEN
                    ELSE
                    ENDIF
                ENDIF
            ENDIF
C****
            IF (ICKFGO .EQ. 1) THEN
                IF (MOD(INCREM,ICKNTR) .EQ. 0) THEN
                    IF (IFLAG .EQ. 4) THEN
                        DO 100 K1 = 1, NNEL
                            KP = NOP(K1,ELNUM)
                            ICODE = IAND(ISPB(KP),4)
C
C ---- IF SHELL ROTATIONS ARE ASSEMBLED IN THE LOCAL SHELL COORDINATE
C      SYSTEM THEN RETREIVE THE LOCAL Z-AXIS FROM STORAGE. ELSE
C      EVALUATE THE NORMAL TO THE SHELL MID PLANE BY A CALL TO THE
C      SHNORM ROUTINE.
C
                            IF (ICODE .GT. 0) THEN
                                CALL SHNORM (ELNUM, K1, NNEL, ITYPE,
     1                              VECT(1,1,K1),VECT(1,2,K1),VECT(1,3
     2                              ,K1))
                            ELSE
                                VECT(1,3,K1) = DBLE(SHELLZ(1,KP))
                                VECT(2,3,K1) = DBLE(SHELLZ(2,KP))
                                VECT(3,3,K1) = DBLE(SHELLZ(3,KP))
                                CALL DIRVEC (VECT(1,1,K1),VECT(1,2,K1)
     1                              ,VECT(1,3,K1))
                            ENDIF
  100                   CONTINUE
                    ENDIF
                ENDIF
            ENDIF
C
            IF (IOEL(ELNUM) .NE. 1) GO TO 240
            IF (PROPER(25,MATRL) .EQ. 2) GO TO 220
            WRITE (ICRK, 1000) ELNUM
C
            IF (PROPER(25,MATRL) .EQ. 0) WRITE (ICRK, 1010) LYNUM
            IF (PROPER(25,MATRL) .EQ. 1) WRITE (ICRK, 1020) LYNUM
C
            NIP = NIPXI*NIPETA*NIPSI
            DO 210 L = 1, NIP
                IF (INTGFG .EQ. 0) GO TO 180
                WRITE (ICRK, 1320) L
C
                IF (PROPER(25,MATRL) .EQ. 0) THEN
C
C ... STEEL LAYER OUTPUT THE RELEVENT STRESS/STRAIN(PRIN.), STIFFNESS
C     AND OTHER STATE VARIABLES
C
                    IF (L .NE. 5) GO TO 200
                    WRITE (ICRK, 900) HISTO(L,15,LYNUM)
                    WRITE (ICRK, 910) HISTO(L,16,LYNUM)
                    WRITE (ICRK, 920) HISTO(L,1,LYNUM)
                    WRITE (ICRK, 930) HISTO(L,11,LYNUM)
                    SIGSTL = HISTO(L,1,LYNUM)*HISTO(L,11,LYNUM)
                    WRITE (ICRK, 940) SIGSTL
                    WRITE (ICRK, 950) HISTO(L,14,LYNUM)
                    WRITE (ICRK, 960) HISTO(L,12,LYNUM)
```

```
                WRITE (ICRK, 970) HISTO(L,13,LYNUM)
                IF (HISTO(L,16,LYNUM).EQ.2 .OR. HISTO(L,16,LYNUM)
     1              .EQ.3) WRITE (ICRK, 980) HISTO(L,10,LYNUM)
                DO IK = 1, 3
                    WRITE (ICRK, 990) (DCS(IK,JK), JK = 1, 3)
                END DO
C
            ELSE IF (PROPER(25,MATRL) .EQ. 1) THEN
                IF (L.NE.1 .AND. L.NE.2 .AND. L.NE.3 .AND. L.NE.4
     1              ) GO TO 200
C
                WRITE (ICRK, 1030)
                WRITE (ICRK, 900) HISTO(L,29,LYNUM)
                WRITE (ICRK, 910) HISTO(L,30,LYNUM)
                WRITE (ICRK, 920) HISTO(L,20,LYNUM)
                WRITE (ICRK, 1040) HISTO(L,21,LYNUM)
                WRITE (ICRK, 1050) HISTO(L,25,LYNUM)
                WRITE (ICRK, 1060) HISTO(L,54,LYNUM)
                WRITE (ICRK, 1070) HISTO(L,22,LYNUM)
                WRITE (ICRK, 1070) HISTO(L,23,LYNUM)
                WRITE (ICRK, 1070) HISTO(L,24,LYNUM)
                IF (HISTO(L,61,LYNUM) .GE. PROPER(6,MATRL)) WRITE
     1              (ICRK, *)
     2              ' UNIAXIAL STRENGTH EXCEEDED FT',
     3              PROPER(6,MATRL)
                WRITE (ICRK, 1080) HISTO(L,61,LYNUM)
                WRITE (ICRK, 1080) HISTO(L,62,LYNUM)
                WRITE (ICRK, 1080) HISTO(L,63,LYNUM)
C
                IF (HISTO(L,30,LYNUM).EQ.2 .OR. HISTO(L,30,LYNUM)
     1              .EQ.3) THEN
                    WRITE (ICRK, 1120) HISTO(L,31,LYNUM)
                    WRITE (ICRK, 1120) HISTO(L,32,LYNUM)
                    WRITE (ICRK, 1120) HISTO(L,33,LYNUM)
                ENDIF
                IF (HISTO(L,29,LYNUM).EQ.2 .OR. HISTO(L,29,LYNUM)
     1              .EQ.3) THEN
                    WRITE (ICRK, 1110) HISTO(L,34,LYNUM)
                    WRITE (ICRK, 1110) HISTO(L,35,LYNUM)
                    WRITE (ICRK, 1110) HISTO(L,36,LYNUM)
                    IF (PROPER(23,MATRL) .EQ. 1) THEN
                        WRITE (ICRK, 1370) HISTO(L,57,LYNUM)
                        WRITE (ICRK, 1380) HISTO(L,59,LYNUM)
                    ENDIF
                ENDIF
C
                DO KK = 10, 18, 3
                    WRITE (ICRK, 990) HISTO(L,KK,LYNUM), HISTO(L,
     1                  KK+1,LYNUM), HISTO(L,KK+2,LYNUM)
                END DO
C
                IF (CRACK(L,1,LYNUM) .GE. 1.0) THEN
                    WRITE (ICRK, 1090) HISTO(L,41,LYNUM)
                    WRITE (ICRK, 1090) HISTO(L,42,LYNUM)
                    WRITE (ICRK, 1090) HISTO(L,43,LYNUM)
                    IF (HISTO(L,30,LYNUM).EQ.2 .OR. HISTO(L,30,
     1                  LYNUM).EQ.3) THEN
                        WRITE (ICRK, 1100) HISTO(L,38,LYNUM)
                        WRITE (ICRK, 1100) HISTO(L,39,LYNUM)
                        WRITE (ICRK, 1100) HISTO(L,40,LYNUM)
                    ENDIF
                    WRITE(ICRK,1130)(HISTO(L,KK,LYNUM),KK=44,46)
                ENDIF
```

```
                IF (CRACK(L,1,LYNUM) .GT. 0.0) THEN
                    NCRACK = INT(CRACK(L,1,LYNUM))
                    WRITE (ICRK, 1350) NCRACK
                    DO 140 ICRACK = 1, NCRACK
                        WRITE (ICRK, 1140) ICRACK
                        WRITE (ICRK, 1150) CRACK(L,117+ICRACK,
     1                      LYNUM)
                        WRITE(ICRK,1160)CRACK(L,37+ICRACK,LYNUM)
                        WRITE(ICRK,1170)CRACK(L,57+ICRACK,LYNUM)
                        WRITE(ICRK,1180)CRACK(L,67+ICRACK,LYNUM)
                        WRITE(ICRK,1180)CRACK(L,77+ICRACK,LYNUM)
                        WRITE(ICRK,1190)CRACK(L,87+ICRACK,LYNUM)
                        WRITE(ICRK,1200)CRACK(L,97+ICRACK,LYNUM)
                        WRITE (ICRK, 1200) CRACK(L,107+ICRACK,
     1                      LYNUM)
                        WRITE (ICRK, 1210) CRACK(L,127+ICRACK,
     1                      LYNUM)
                        ICK = 9*(ICRACK-1) + 1
                        DO KKK = ICK, ICK + 8, 3
                            WRITE (ICRK, 1220) CRACK(L,137+KKK,
     1                          LYNUM), CRACK(L,138+KKK,LYNUM),
     2                          CRACK(L,139+KKK,LYNUM)
                        END DO
                        IF (CRACK(L,117+ICRACK,LYNUM).EQ.2.0 .OR.
     1                      CRACK(L,117+ICRACK,LYNUM).EQ.3.0)
     2                      WRITE (ICRK, 1230) CRACK(L,47+ICRACK,
     3                      LYNUM)
 140                CONTINUE
                    DO 170 ISP = 1, 3
                        IF (TENSTF(L,ISP,LYNUM) .EQ. 0) GO TO 160
                        WRITE (ICRK, 1360) ISP
                        WRITE (ICRK, 1240) TENSTF(L,21+ISP,LYNUM)
                        WRITE (ICRK, 1250) TENSTF(L,6+ISP,LYNUM)
                        WRITE (ICRK, 1260) TENSTF(L,18+ISP,LYNUM)


                        WRITE (ICRK, 1270) TENSTF(L,24+ISP,LYNUM)
                        WRITE (ICRK, 1280) TENSTF(L,9+ISP,LYNUM)
                        WRITE (ICRK, 1290) TENSTF(L,12+ISP,LYNUM)
                        ICM = 9*(ISP-1) + 1
                        DO KKK = ICM, ICM + 8, 3
                            WRITE (ICRK, 1300) TENSTF(L,30+KKK,
     1                          LYNUM), TENSTF(L,31+KKK,LYNUM),
     2                          TENSTF(L,32+KKK,LYNUM)
                        END DO
                        IF (TENSTF(L,21+ISP,LYNUM).EQ.2 .OR.
     1                      TENSTF(L,21+ISP,LYNUM).EQ.3) WRITE (
     2                      ICRK, 1310) TENSTF(L,15+ISP,LYNUM)
 160                    CONTINUE
 170                CONTINUE
                ENDIF
            ENDIF
C
C* THIS PORTION IS TO OBTAIN CRACK ORIENTATION PLANE FOR POST PROCESSING
C
 180        CONTINUE
            IF (PROPER(25,MATRL) .EQ. 1.0) THEN
                IF (ICKFGO .EQ. 1) THEN
                    IF (MOD(INCREM,ICKNTR) .EQ. 0) THEN
                        IF (CRACK(L,1,LYNUM) .GE. 1.0) THEN
                            X1 = 0.0
                            Y1 = 0.0
                            Z1 = 0.0
                            DETJAC = 0.0
```

```
                              SIZE = 0.0
                              THICKE = 0.0
                              RAD = 0.0
                              IF (IFLAG .EQ. 0) THEN
                              CALL JACB3D (L, ELNUM, NNEL, IERROR,
     1                            DETJAC)
                              CALL COORD1 (ELNUM, NNEL, L, ITYPE, X1
     1                            , Y1, Z1)
C
                              SIZE = (W(L)*DABS(DETJAC))**(1./3.)
C
                              ELSE IF (IFLAG .EQ. 4) THEN
                              CALL GETTHK(L,ELNUM,NNEL,THICKE,RAD)
C
C ----- VECTOR V3 RETURNED BY JACSHL IS NORMAL TO MID SURFACE OF THE
C       SHELL AT INTEGRATION POINTS
C
                              ISET = 0
                              SIP = SI(L)
                              CALL JACSHL (L, ELNUM, NNEL, THICKE,
     1                            DETJAC, V3, ISET, SIP)
                              CALL COORD1 (ELNUM, NNEL, L, ITYPE, X1
     1                            , Y1, Z1)
C
                              SIZE = (W(L)*DABS(DETJAC))**(1./3.)
C
                              ENDIF
C
                              NCRACK = INT(CRACK(L,1,LYNUM))
                              DO KP = 1, NCRACK
                              KK = 9*(KP-1) + 1
                              AL1 = CRACK(L,137+KK,LYNUM)
                              AM1 = CRACK(L,138+KK,LYNUM)
                              AN1 = CRACK(L,139+KK,LYNUM)
                              AL2 = CRACK(L,140+KK,LYNUM)
                              AM2 = CRACK(L,141+KK,LYNUM)
                              AN2 = CRACK(L,142+KK,LYNUM)
                              AL3 = CRACK(L,143+KK,LYNUM)
                              AM3 = CRACK(L,144+KK,LYNUM)
                              AN3 = CRACK(L,145+KK,LYNUM)
                              ISTAT = 1
                              IF (CRACK(L,117+KP,LYNUM) .EQ. 1)
     1                            ISTAT = 1
                              IF (CRACK(L,117+KP,LYNUM) .EQ. 2)
     1                            ISTAT = 2
                              IF (CRACK(L,117+KP,LYNUM) .EQ. 3)
     1                            ISTAT = 2
                              XA = X1 + AL2*SIZE/2.0
                              YA = Y1 + AM2*SIZE/2.0
                              ZA = Z1 + AN2*SIZE/2.0
                              XB = X1 + ((-AL2*SIZE/2.0))
                              YB = Y1 + ((-AM2*SIZE/2.0))
                              ZB = Z1 + ((-AN2*SIZE/2.0))
                              XC = X1 + AL3*SIZE/2.0
                              YC = Y1 + AM3*SIZE/2.0
                              ZC = Z1 + AN3*SIZE/2.0
                              XD = X1 + ((-AL3*SIZE/2.0))


                              YD = Y1 + ((-AM3*SIZE/2.0))
                              ZD = Z1 + ((-AN3*SIZE/2.0))
C
                              WRITE (ICK2, 1390) ILINE, ISTAT
                              WRITE (ICK2, 1400)
```

```
                              WRITE (ICK2, 1410)
                              WRITE (ICK2, 1420) XA, YA, ZA
                              WRITE (ICK2, 1420) XC, YC, ZC
                              WRITE (ICK2, 1390) ILINE + 1, ISTAT
                              WRITE (ICK2, 1400)
                              WRITE (ICK2, 1410)
                              WRITE (ICK2, 1420) XC, YC, ZC
                              WRITE (ICK2, 1420) XB, YB, ZB
                              WRITE (ICK2, 1390) ILINE + 2, ISTAT
                              WRITE (ICK2, 1400)
                              WRITE (ICK2, 1410)
                              WRITE (ICK2, 1420) XB, YB, ZB
                              WRITE (ICK2, 1420) XD, YD, ZD
                              WRITE (ICK2, 1390) ILINE + 3, ISTAT
                              WRITE (ICK2, 1400)
                              WRITE (ICK2, 1410)
                              WRITE (ICK2, 1420) XD, YD, ZD
                              WRITE (ICK2, 1420) XA, YA, ZA
                              ILINE = ILINE + 4
                              END DO
                            ENDIF
                          ENDIF
                        ENDIF
                      ENDIF
 200                  CONTINUE
 210                CONTINUE
 220                CONTINUE
 230              CONTINUE
 240              CONTINUE
 250  CONTINUE
      IF (ICKFGO .EQ. 1) THEN
C     WRITE(*,*)'ICKFGO',ICKFGO,'INCREM',INCREM
        IF (MOD(INCREM,ICKNTR) .EQ. 0) WRITE (ICK2, '(A6)') '    -1'
      ENDIF
      RETURN
 900  FORMAT(10X,D12.5,'IS THE STRESS POINT STATUS')
 910  FORMAT(10X,D12.5,'IS THE LOADING STATUS')
 920  FORMAT(10X,D12.5,'IS THE CURRENT SECANT STIFFNESS')
 930  FORMAT(10X,D12.5,'IS THE CURRENT STRAIN ')
 940  FORMAT(10X,D12.5,'IS THE CURRENT STRESS ')
 950  FORMAT(10X,D12.5,'IS THE INITIAL STIFFNESS')
 960  FORMAT(10X,D12.5,'IS THE STRAIN AT YEILD')
 970  FORMAT(10X,D12.5,'IS THE STRESS AT YEILD')
 980  FORMAT(10X,D12.5,'IS THE STRAIN AT UN/RELOAD')
 990  FORMAT(10X,3(2X,D12.5),'ARE THE DIR. COSINES ')
1000  FORMAT(1X,I5,'************IS THE ELEMENT NO.','************')
1010  FORMAT(1X,I5,'***********IS THE STEEL LAYER NO.','************')
1020  FORMAT(1X,I5,'*********IS THE CONCRETE LAYER NO.','************')
1030  FORMAT(1X,'***********INTACT CONCRETE INFORMATION','************')
1040  FORMAT(10X,D12.5,'IS THE CURRENT SECANT POISSONS RATIO')
1050  FORMAT(10X,D12.5,'IS THE CURRENT STRESS STATE REGION')
1060  FORMAT(10X,D12.5,'IS THE FAILURE FUNCTION VALUE')
1070  FORMAT(10X,D12.5,'IS THE PRINCIPAL STRAIN INTACT CONCRETE VALUE')
1080  FORMAT(10X,D12.5,'IS THE PRINCIPAL STRESS VALUE')
1090  FORMAT(10X,D12.5,'IS THE TOTAL PRINCIPAL STRAIN VALUE')
1100  FORMAT(10X,D12.5,'TOTAL PRINCIPAL STRAIN UN/RELOAD VALUE')
1110  FORMAT(10X,D12.5,'PRINCIPAL STRESS VALUE AT FAILURE')
1120  FORMAT(10X,D12.5,'PRINCIPAL STRAIN UN/RELOAD VALUE')
1130  FORMAT(10X,3(2X,D12.5),'DIR. COSINE CRACKED CONCRETE ')
1140  FORMAT(10X,I5,'CURRENT CRACK NO.')
1150  FORMAT(10X,D12.5,'IS THE STATE OF THE CRACK')
1160  FORMAT(10X,D12.5,'IS THE CRACK INITIATION STRESS')
1170  FORMAT(10X,D12.5,'CRACK NORMAL STRAIN CURRENT')
```

```
 1180 FORMAT(10X,D12.5,'IS THE CRACK SHEAR STRAIN CURRENT')
 1190 FORMAT(10X,D12.5,'IS THE CRACK NORMAL STIFFNESS')
 1200 FORMAT(10X,D12.5,'IS THE CRACK SHEAR STIFFNESS')
 1210 FORMAT(10X,D12.5,'IS THE CRACK TERM. STRAIN')
 1220 FORMAT(10X,3(2X,D12.5),'ARE DIR. COSINE OF CRACK')
 1230 FORMAT(10X,D12.5,'IS THE CRACK UNLOADING STRAIN')
 1240 FORMAT(10X,D12.5,'IS THE STATE OF TEN. STIF. SPG.')
 1250 FORMAT(10X,D12.5,'IS THE INITIATION STRESS ')
 1260 FORMAT(10X,D12.5,'IS THE CURRENT STRAIN')
 1270 FORMAT(10X,D12.5,'IS THE CURRENT STIFFNESS ')
 1280 FORMAT(10X,D12.5,'IS THE STRAIN AT INITIATION')
 1290 FORMAT(10X,D12.5,'IS THE STRAIN AT TERMINATION')
 1300 FORMAT(10X,3(2X,D12.5),'ARE DIR. COSINE OF SPRING')
 1310 FORMAT(10X,D12.5,'IS THE STRAIN AT UN/RELOADING')
 1320 FORMAT(1X,I5,'*****IS THE INTEGRATION POINT #.','************',/)
 1330 FORMAT(1X,I5,'    INCREMENT NO.','         ')
 1340 FORMAT(1X,I5,'    ITTERATION NO.','      ')
 1350 FORMAT(1X,/,I5,'*CRACK INFORMATION---NO.# OF CRACKS','*',/)
 1360 FORMAT(1X,/,I5,'*TENSION STIFFENING SPRING NO.','*',/)
 1370 FORMAT(10X,D12.5,'IS THE SECANT STIFFNESS AT FAILURE')
 1380 FORMAT(10X,D12.5,'IS THE POISSONS RATIO AT FAILURE')
 1390 FORMAT(I10,9X,'0',9X,'1',9X,'3',I10,9X,'1',9X,'0',9X,'0')
 1400 FORMAT(9X,'0',9X,'0',9X,'0',9X,'1',9X,'0')
 1410 FORMAT(5X,'1',5X,'1')
 1420 FORMAT(1P,3D25.16)
 1430 FORMAT(I10)
 1440 FORMAT((A80))
      END
C
C ======================= O U T P U T ============================
C
      SUBROUTINE OUTPUT(VALUE,IOUT,IERROR)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C OUT1      OUT2      OUT3      OUT4      OUT7      OUT10
C OUT8      OUT9      OUT28     OUT20     OUT21     OUT22
C
C***********************************************************
      REAL*8 VALUE( * )
      COMMON/OUTPT1/IOFLAG,IFCAED
C
C ---- DESCRIPTION OF THE OUTPUT MODULES AND THE CORRESPONDING BIT
C       FLAG POSITION IN THE IOFLAG INTEGER VARIABLE
C
C       FILE    BIT  DESCRIPTION
C
C       OUT1    1    OUTPUT ELEMENT STRESSES AT THE INTEGRATION POINTS
C       OUT2    2    OUTPUT ELEMENT STRAINS AT THE INTEGRATION POINTS
C       OUT3    3    OUTPUT ELEMENT STRESSES AT THE ELEMENT NODES
C       OUT4    4    OUTPUT ELEMENT STRAINS AT THE ELEMENT NODES
C       OUT5    5    OUTPUT AVERAGE STRESSES AT THE NODES
C       OUT6    6    OUTPUT AVERAGE STRAINS AT THE NODES
C       OUT7    7    OUTPUT DISPLACEMENTS AT THE NODES
C       OUT8    8    OUTPUT NODAL EQUILIBRIUM FORCES AT THE NODES
C       OUT9    9    OUTPUT NODAL EQUILIBRIUM FORCES FOR FREEBODIES
C       OUT10   10   OUTPUT REACTIONS AT THE SUPPORTS
C       OUT11   11   OUTPUT NODAL COORDINATES
C       OUT12   12   OUTPUT ELEMENT CONNECTIVITY
```

```
C
C ---- THE FOLLOWING ARE CAEDS UNIVERSAL DATASETS
C
C       OUT20   20   OUTPUT SECOND PIOLA KIRCHHOFF STRESSES AT THE
C                    ELEMENT NODES
C               21   OUTPUT CAUCHY STRESSES AT THE ELEMENT NODES
C               22   OUTPUT TOTAL ELEMENT STRAINS AT THE ELEMENT NODES
C               23   OUTPUT ELEMENT ELASTIC STRAINS AT THE ELEMENT NODE
C               24   OUTPUT PLASTIC ELEMENT STRAINS AT THE ELEMENT NODE
C       OUT21   25   OUTPUT NODAL DISPLACEMENTS
C       OUT22   26   OUTPUT SUPPORT REACTIONS
C       OUT23   27   OUTPUT NODAL EQUILIBRIUM LOADS
C
      IF (BTEST(IOFLAG,1)) CALL OUT1 (IOUT)
      IF (BTEST(IOFLAG,2)) CALL OUT2 (IOUT)
      IF (BTEST(IOFLAG,3)) CALL OUT3 (IOUT)
      IF (BTEST(IOFLAG,4)) CALL OUT4 (IOUT)
      IF (BTEST(IOFLAG,7)) CALL OUT7 (IOUT)
      IF (BTEST(IOFLAG,10)) CALL OUT10 (IOUT)
      IF (BTEST(IOFLAG,8)) CALL OUT8 (IOUT)
      IF (BTEST(IOFLAG,8)) CALL OUT9 (IOUT)
      IF (BTEST(IOFLAG,28)) CALL OUT28 (IOUT)
C
      IFCAED = 1
      IF (1 .EQ. 1) THEN
          IF (BTEST(IOFLAG,20)) CALL OUT20 (IOUT, 1)
          IF (BTEST(IOFLAG,22)) CALL OUT20 (IOUT, 3)
          IF (BTEST(IOFLAG,25)) CALL OUT21 (IOUT)
          IF (BTEST(IOFLAG,26)) CALL OUT22 (IOUT)
      ENDIF
C
      RETURN
      END
C
C ========================== O U T 1 ==============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE OUT1(IOUT)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ELINFO    ELINTM    LYINFO    ISH3DG    ISHSHL    COORD1
C IOGET     REWIN
C
C***********************************************************
      REAL*4 THICK
      REAL*8 LTHICK
      CHARACTER*57 CTEMP
      INTEGER ELNUM
      COMMON/UTIL1/STRESS(6),STRAIN(6),STRELA(6),CTEMP
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1              NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC1(10),LDCONT,INCPTR
      COMMON/INPUTF/MATYPE(10)
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/INPUT9/THICK(9),IFLAG
      COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
      COMMON/OUTPT2/IOEL(5000),IONOD(10000)
```

```
      DIMENSION COORDS(3),CSTR(6)
C
      IEND = 0
      DO 130 ELNUM = 1, NELEM
          CALL ELINFO (ELNUM, ITYPE, NNEL, IFLAG, ISTART, LINES)
          CALL ELINTM (ELNUM, IDENT, INTCOD, NIPXI, NIPETA, NIPSI,
     1        MATNUM, THICK)
          IF (IOEL(ELNUM) .EQ. 1) WRITE (IOUT, 910) ELNUM
C
          DO 110 LYNUM = 1, NLAYRS
              IF (IOEL(ELNUM) .EQ. 1) WRITE (IOUT, 920) LYNUM
              CALL LYINFO (LYNUM, LTHICK, ZS, MATRL, DCS, NNEL, ELNUM,
     1            NIPXI, NIPETA, NIPSI)
C
              IF (ITYPE .LE. 0) GO TO 120
              IF (ITYPE.NE.0 .OR. IDENT.NE.0) THEN
                  IF (ITYPE .GT. 300) THEN
                      ASSIGN 980 TO IFOR
                      ASSIGN 990 TO IFOR1
                      IF (IFLAG1 .EQ. 1) THEN
                          ASSIGN 1010 TO IF2
                      ELSE
                          ASSIGN 1000 TO IF2
                      ENDIF
                      IF (IFLAG .EQ. 0) THEN
                          CALL ISH3DG (ITYPE, NNEL, IERROR)
                      ELSE IF (IFLAG .EQ. 4) THEN
                          CALL ISHSHL (ITYPE, NNEL, IERROR)
                      ENDIF
                      IEND = 6
                  ENDIF
              ENDIF
C     ITYPE1 = ITYPE
C     IDENT1 = IDENT
C
              IF (IOEL(ELNUM) .EQ. 1) WRITE (IOUT, IF2)
              DO 100 INTGPN = 1, NIP
                  CALL COORD1 (ELNUM, NNEL, INTGPN, ITYPE, COORDS(1),
     1                COORDS(2),COORDS(3))
                  CALL IOGET (LDEV1, 96, '(A96)', 5)
                  IF (IOEL(ELNUM) .EQ. 1) WRITE (IOUT, IFOR) INTGPN, (
     1                COORDS(K1), K1 = 1, IDIM), (STRESS(K1), K1 = 1,
     2                IEND)
  100         CONTINUE
C
  110     CONTINUE
C
  120     CONTINUE
  130 CONTINUE
      CALL REWIN
C
      RETURN
C
  900 FORMAT(I6,1P,6E14.5)
  910 FORMAT(/,20X,'S T R E S S E S   AT   I N T E G R A T I O N   ',
     1       'P O I N T S   O N   E L E M E N T',I7)
  920 FORMAT(/,20X,'S T R E S S E S   AT   I N T E G R A T I O N   ',
     1       'P O I N T S   O F   L A Y E R',I7)
  930 FORMAT(34X,1P,4E14.5/)
C
  940 FORMAT(1X,'POINT',13X,'X',13X,'Y',
     1       11X,'SXX',11X,'SYY',11X,'SXY',11X,'SZZ'/)
C
```

```
  950 FORMAT(1X,'POINT',13X,'X',13X,'Y',
     1      3X,'2ND PIOLA_X',3X,'2ND PIOLA_Y',2X,'2ND PIOLA_XY',3X,
     2      '2ND_PIOLA_Z'/
     3      40X,'CAUCHY_X',6X,'CAUCHY_Y',6X,'CAUCHY_XY',5X,'CAUCHY_Z'/)
C
  960 FORMAT(1X,'POINT',13X,'X',13X,'Y',
     1       11X,'SR ',11X,'SY ',11X,'SRY',11X,' ST'/)
C
  970 FORMAT(1X,'POINT',13X,'X',13X,'Y',
     1       3X,'2ND PIOLA_R',3X,'2ND PIOLA_Y',3X,'2ND PIOLA_RY',2X,
     2       '2ND_PIOLA_T'/
     3      40X,'CAUCHY_R',6X,'CAUCHY_Y',6X,'CAUCHY_RY',5X,'CAUCHY_T'/)
C
  980 FORMAT(I6,1P,9E14.5)
  990 FORMAT(48X,1P,6E14.5/)
C
 1000 FORMAT(1X,'POINT',13X,'X',13X,'Y',13X,
     1'Z',11X,'SXX',11X,'SYY',11X,'SZZ',11X,'SXY',11X,'SYZ',11X,'SXZ'/)
C
 1010 FORMAT(1X,'POINT',13X,'X',13X,'Y',
     1   13X,'Z',3X,'2ND PIOLA_X',3X,'2ND PIOLA_Y',3X,'2ND PIOLA_Z',2X,
     2   '2ND_PIOLA_XY',2X,'2ND PIOLA_YZ',2X,'2ND PIOLA_XZ'/
     3   54X,'CAUCHY_X',6X,'CAUCHY_Y',6X,'CAUCHY_Z',5X,'CAUCHY_XY',5X,
     4   'CAUCHY_YZ',5X,'CAUCHY_XZ'/)
C
      END


C
C ========================== O U T 2 ================================
C
C     INCLUDE (PROCESS)
      SUBROUTINE OUT2(IOUT)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ELINFO    ELINTM    LYINFO    ISH3DG    ISHSHL    IOGET
C COORD1    REWIN
C
C***************************************************************
      REAL*4 THICK
      REAL*8 LTHICK
      CHARACTER*57 CTEMP
      INTEGER ELNUM
      COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
      COMMON/UTIL1/STRESS(6),STRAIN(6),STRELA(6),CTEMP
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1             NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC1(10),LDCONT,INCPTR


      COMMON/INPUTF/MATYPE(10)
      COMMON/INPUT9/THICK(9),IFLAG
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/OUTPT2/IOEL(5000),IONOD(10000)
      DIMENSION COORDS(3),STRPLA(6)
C
      IEND = 0
      DO 130 ELNUM = 1, NELEM
          CALL ELINFO (ELNUM, ITYPE, NNEL, IFLAG, ISTART, LINES)
```

```
         CALL ELINTM (ELNUM, IDENT, INTCOD, NIPXI, NIPETA, NIPSI,
     1        MATNUM, THICK)
C
         IF (IOEL(ELNUM) .EQ. 1) WRITE (IOUT, 910) ELNUM
         DO 110 LYNUM = 1, NLAYRS
            IF (IOEL(ELNUM) .EQ. 1) WRITE (IOUT, 920) LYNUM
            CALL LYINFO (LYNUM, LTHICK, ZS, MATRL, DCS, NNEL, ELNUM,
     1           NIPXI, NIPETA, NIPSI)
C
            IF (ITYPE .LE. 0) GO TO 120
            IF (ITYPE.NE.0 .OR. IDENT.NE.0) THEN
               IF (ITYPE .GT. 300) THEN
                  ASSIGN 980 TO IFOR
                  ASSIGN 990 TO IFOR1
                  IF (MATYPE(MATNUM) .EQ. 2) THEN
                     ASSIGN 1010 TO IF1
                  ELSE
                     ASSIGN 1000 TO IF1
                  ENDIF
                  IF (IFLAG .EQ. 0) THEN
                     CALL ISH3DG (ITYPE, NNEL, IERROR)
                  ELSE IF (IFLAG .EQ. 4) THEN
                     CALL ISHSHL (ITYPE, NNEL, IERRORS)
                  ENDIF
                  IEND = 6
               ENDIF
            ENDIF
C     ITYPE1 = ITYPE
C     IDENT1 = IDENT
C
            IF (IOEL(ELNUM) .EQ. 1) WRITE (IOUT, IF1)
            DO 100 INTGPN = 1, NIP
               CALL IOGET (LDEV1, 96, '(A96)', 5)
               CALL COORD1 (ELNUM, NNEL, INTGPN, ITYPE, COORDS(1),
     1              COORDS(2),COORDS(3))
               IF (IOEL(ELNUM) .EQ. 1) WRITE (IOUT, IFOR) INTGPN, (
     1              COORDS(K1), K1 = 1, IDIM), (STRAIN(K1), K1 = 1,
     2              IEND)
  100       CONTINUE
C
  110    CONTINUE
  120    CONTINUE
  130 CONTINUE
C
      CALL REWIN
      RETURN
C
  900 FORMAT(I6,1P,6E14.5)
  910 FORMAT(/,20X,'S T R A I N S   AT   I N T E G R A T I O N   ',
     1       'P O I N T S   O N   E L E M E N T',I7)
  920 FORMAT(/,20X,'S T R A I N S   AT   I N T E G R A T I O N   ',
     1       'P O I N T S   O N   L A Y E R',I7)
C
  930 FORMAT(2(34X,1P,4E14.5/))
C
  940 FORMAT(1X,'POINT',13X,'X',13X,'Y',
     111X,'EXX',11X,'EYY',11X,'EXY',11X,'EZZ'/)
C
  950 FORMAT(1X,'POINT',13X,'X',13X,'Y',
     16X,' TOTAL_X    ',3X,' TOTAL_Y    ',3X,' TOTAL_XY    ',2X,
     2' TOTAL_Z    '/
     340X,' ELAST_X',6X,' ELAST_Y',6X,' ELAST_XY',5X,' ELAST_Z'/
     440X,' PLAST_X',6X,' PLAST_Y',6X,' PLAST_XY',5X,' PLAST_Z'/)
```

```
C
  960 FORMAT(1X,'POINT',5X,'X',14X,'Y',
     111X,'ER ',11X,'EY ',11X,'ERY',11X,'ET '/)
C
  970 FORMAT(1X,'POINT',13X,'X',13X,'Y',
     16X,' TOTAL_R   ',3X,' TOTAL_Y   ',3X,' TOTAL_RY   ',2X,
     2' TOTAL_T   '/
     340X,' ELAST_R',6X,' ELAST_Y',6X,' ELAST_RY',5X,' ELAST_T'/
     440X,' PLAST_R',6X,' PLAST_Y',6X,' PLAST_RY',5X,' PLAST_T'/)
C
  980 FORMAT(I6,1P,9E14.5)
  990 FORMAT(2(48X,1P,6E14.5/))
C
 1000 FORMAT(1X,'POINT',13X,'X',13X,'Y',13X,
     1'Z',11X,'EXX',11X,'EYY',11X,'EZZ',11X,'EXY',11X,'EYZ',11X,'EXZ'/)
C
 1010 FORMAT(1X,'POINT',13X,'X',13X,'Y',
     113X,'Z',6X,' TOTAL_X   ',3X,' TOTAL_Y   ',3X,' TOTAL_Z   ',2X,
     2' TOTAL_XY   ',2X,' TOTAL_YZ   ',2X,' TOTAL_XZ   '/

     354X,' ELAST_X',6X,' ELAST_Y',6X,' ELAST_Z',5X,' ELAST_XY',5X,
     4' ELAST_YZ',5X,' ELAST_XZ'/
     554X,' PLAST_X',6X,' PLAST_Y',6X,' PLAST_Z',5X,' PLAST_XY',5X,
     6' PLAST_YZ',5X,' PLAST_XZ'/)
C
      END
C

C ========================= O U T 3 ==============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE OUT3(IOUT)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ELINFO    ELINTM    LYINFO    LAGRG1    IOGET     REWIN
C
C***********************************************************
      REAL*4 THICK
      REAL*8 LTHICK
      CHARACTER*57 CTEMP
      INTEGER ELNUM
      COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
      COMMON/UTIL1/STRESS(6),STRAIN(6),STRELA(6),CTEMP
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/INPUT2/NOP(20,5000)
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1              NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC1(10),LDCONT,INCPTR
      COMMON/INPUTF/MATYPE(10)
      COMMON/ISPC01/P(64 , 27)
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/INPUT9/THICK(9),IFLAG
      COMMON/OUTPT2/IOEL(5000),IONOD(10000)
      DIMENSION CAUC(6,27),CAUCH(6),STRS(6,27)
C
      IEND = 0
      DO 180 ELNUM = 1, NELEM
         CALL ELINFO (ELNUM, ITYPE, NNEL, IFLAG, ISTART, LINES)
```

```
          CALL ELINTH (ELNUM, IDENT, INTCOD, NIPXI, NIPETA, NIPSI,
     1        MATNUM, THICK)
C
          DO 160 LYNUM = 1, NLAYRS
              CALL LYINFO (LYNUM, LTHICK, ZS, MATRL, DCS, NNEL, ELNUM,
     1            NIPXI, NIPETA, NIPSI)
C
              IF (ITYPE .LE. 0) GO TO 170
              IF (ITYPE.NE.0 .OR. IDENT.NE.0) THEN
                  CALL LAGRG1 (ITYPE, IFLAG, NNEL, IERROR)
                  IF (ITYPE .GT. 300) THEN
                      ASSIGN 980 TO IFOR
                      ASSIGN 990 TO IFOR1
                      ASSIGN 1000 TO IF2
                      IEND = 6
                  ENDIF
              ENDIF
C
              DO 110 K1 = 1, NNEL
                  DO 100 K2 = 1, IEND
                      CAUC(K2,K1) = 0.
                      STRS(K2,K1) = 0.
  100             CONTINUE
  110         CONTINUE
C
              DO 140 INTGPN = 1, NIP
                  CALL IOGET (LDEV1, 96, '(A96)', 5)
C
                  DO 130 K1 = 1, NNEL
                      DO 120 K2 = 1, IEND
                          STRS(K2,K1) = STRS(K2,K1) + STRESS(K2)*P(
     1                        INTGPN,K1)
  120                 CONTINUE
  130             CONTINUE
  140         CONTINUE
C
              IF (IOEL(ELNUM) .EQ. 1) WRITE (IOUT, 910) ELNUM
              IF (IOEL(ELNUM) .EQ. 1) WRITE (IOUT, 920) LYNUM
              IF (IOEL(ELNUM) .EQ. 1) WRITE (IOUT, IF2)
              DO 150 K1 = 1, NNEL
                  IF (IOEL(ELNUM) .EQ. 1) WRITE (IOUT, IFOR) K1, NOP(K1,
     1                ELNUM), (STRS(I,K1), I = 1, IEND)
  150         CONTINUE
C
  160     CONTINUE
C
  170     CONTINUE
  180 CONTINUE
      CALL REWIN
C
      RETURN
C
  900 FORMAT(I6,I10,1P,4E14.5)
  910 FORMAT(/,20X,'S T R E S S E S   AT   T H E   N O D E S   ',
     1       'O N   E L E M E N T',I7)
  920 FORMAT(/,20X,'S T R E S S E S   O F   T H E   L A Y E R #',
     1       'A T  N O D E  L E V E L ',I7)
  930 FORMAT(16X,1P,4E14.5/)
C
  940 FORMAT(1X,'NODE',3X,'NODE ID.',
     1       11X,'SXX',11X,'SYY',11X,'SXY',11X,'SZZ'/)
C
  950 FORMAT(1X,'NODE',3X,'NODE ID.',
```

```
     1       3X,'2ND PIOLA_X',3X,'2ND PIOLA_Y',2X,'2ND PIOLA_XY',3X,
     2       '2ND_PIOLA_Z'/
     3       22X,'CAUCHY_X',6X,'CAUCHY_Y',5X,'CAUCHY_XY',6X,'CAUCHY_Z'/)
C
  960 FORMAT(1X,'NODE',3X,'NODE ID.',
     1        11X,'SR ',11X,'SY ',11X,'SRY',11X,' ST'/)
C
  970 FORMAT(1X,'NODE',3X,'NODE ID.',
     1        3X,'2ND PIOLA_R',3X,'2ND PIOLA_Y',3X,'2ND PIOLA_RY',2X,
     2        '2ND_PIOLA_T'/
     3       22X,'CAUCHY_R',6X,'CAUCHY_Y',5X,'CAUCHY_RY',6X,'CAUCHY_T'/)
C
  980 FORMAT(I5,I11,1P,9E14.5)
  990 FORMAT(16X,1P,6E14.5/)
C
 1000 FORMAT(1X,'NODE',3X,'NODE ID.',
     111X,'SXX',11X,'SYY',11X,'SZZ',11X,'SXY',11X,'SYZ',11X,'SXZ'/)
C
 1010 FORMAT(1X,'NODE',3X,'NODE ID.',
     1  3X,'2ND PIOLA_X',3X,'2ND PIOLA_Y',3X,'2ND PIOLA_Z',2X,
     2  '2ND_PIOLA_XY',2X,'2ND PIOLA_YZ',2X,'2ND PIOLA_XZ'/
     3  22X,'CAUCHY_X',6X,'CAUCHY_Y',6X,'CAUCHY_Z',5X,'CAUCHY_XY',5X,
     4  'CAUCHY_YZ',5X,'CAUCHY_XZ'/)
C
      END
C


C ========================= O U T 4 ==============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE OUT4(IOUT)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ELINFO    ELINTM    LYINFO    LAGRG1    IOGET     REWIN
C
C***********************************************************
      REAL*8 LTHICK
      REAL*4 THICK
      CHARACTER*57 CTEMP
      INTEGER ELNUM
      COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
      COMMON/UTIL1/STRESS(6),STRAIN(6),STRELA(6),CTEMP
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1             NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC1(10),LDCONT,INCPTR
      COMMON/INPUT2/NOP(20,5000)
      COMMON/INPUTF/MATYPE(10)
      COMMON/ISPC01/P(64 , 27)
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/INPUT9/THICK(9),IFLAG
      COMMON/OUTPT2/IOEL(5000),IONOD(10000)
      DIMENSION STRN(6,20),ELSTR(6,20),STRPLA(6)
C
      IEND = 0
      DO 180 ELNUM = 1, NELEM
          CALL ELINFO (ELNUM, ITYPE, NNEL, IFLAG, ISTART, LINES)
          CALL ELINTM (ELNUM, IDENT, INTCOD, NIPXI, NIPETA, NIPSI,
```

```
     1        MATNUM, THICK)
C
          DO 160 LYNUM = 1, NLAYRS
             CALL LYINFO (LYNUM, LTHICK, ZS, MATRL, DCS, NNEL, ELNUM,
     1          NIPXI, NIPETA, NIPSI)
C
             IF (ITYPE .LE. 0) GO TO 170
             IF (ITYPE.NE.0 .OR. IDENT.NE.0) THEN
                CALL LAGRG1 (ITYPE, IFLAG, NNEL, IERROR)
                IF (ITYPE .GT. 300) THEN
                   ASSIGN 980 TO IFOR
                   ASSIGN 990 TO IFOR1
                   ASSIGN 1000 TO IF2
                   IEND = 6
                ENDIF
             ENDIF
C     ITYPE1 = ITYPE
C     IDENT1 = IDENT
C
             DO 110 K1 = 1, NNEL
                DO 100 K2 = 1, IEND
                   STRN(K2,K1) = 0.
                   ELSTR(K2,K1) = 0.
  100           CONTINUE
  110        CONTINUE
C
             DO 140 INTGPN = 1, NIP
                CALL IOGET (LDEV1, 96, '(A96)', 5)
C
                DO 130 K1 = 1, NNEL
                   NODE = NOP(K1,ELNUM)
                   DO 120 K2 = 1, IEND
                      STRN(K2,K1) = STRN(K2,K1) + STRAIN(K2)*P(
     1                   INTGPN,K1)
  120              CONTINUE
  130           CONTINUE
  140        CONTINUE
C
             IF (IOEL(ELNUM) .EQ. 1) WRITE (IOUT, 910) ELNUM
             IF (IOEL(ELNUM) .EQ. 1) WRITE (IOUT, 920) LYNUM
             IF (IOEL(ELNUM) .EQ. 1) WRITE (IOUT, IF2)
             DO 150 K1 = 1, NNEL
                IF (IOEL(ELNUM) .EQ. 1) WRITE (IOUT, IFOR) K1, NOP(K1,
     1             ELNUM), (STRN(I,K1), I = 1, IEND)
  150        CONTINUE
C
  160     CONTINUE
C
  170     CONTINUE
  180 CONTINUE
      CALL REWIN
C
      RETURN
C
  900 FORMAT(I6,I10,1P,4E14.5)
  910 FORMAT(/,20X,'S T R A I N S   AT   T H E   N O D E S   ',
     1       'O N   E L E M E N T',I7)
  920 FORMAT(/,20X,'S T R A I N S   AT   L A Y E R #  ',
     1       'A T  T H E  N O D E  L E V E L',I7)
  930 FORMAT(2(16X,1P,4E14.5/))
C
  940 FORMAT(1X,'NODE',3X,'NODE ID.',
     1       11X,'EXX',11X,'EYY',11X,'EXY',11X,'EZZ'/)
```

```
C
  950 FORMAT(1X,'NODE',3X,'NODE ID.',
     1      3X,'     TOTAL_X',3X,'     TOTAL_Y',2X,'     TOTAL_XY',3X,
     2      '     TOTAL_Z'/
     3      21X,'ELASTIC_X',5X,'ELASTIC_Y',4X,'ELASTIC_XY',5X,
     4      'ELASTIC_Z'/
     5      21X,'PLASTIC_X',5X,'PLASTIC_Y',4X,'PLASTIC_XY',5X,
     6      'PLASTIC_Z'/)
C
  960 FORMAT(1X,'NODE',3X,'NODE ID.',
     1       11X,'ER ',11X,'EY ',11X,'ERY',11X,' ET'/)
C
  970 FORMAT(1X,'NODE',3X,'NODE ID.',
     1      3X,'     TOTAL_R',3X,'     TOTAL_Y',2X,'     TOTAL_RY',3X,
     2      '     TOTAL_T'/
     3      21X,'ELASTIC_R',5X,'ELASTIC_Y',4X,'ELASTIC_RY',5X,
     4      'ELASTIC_T'/
     5      21X,'PLASTIC_R',5X,'PLASTIC_Y',4X,'PLASTIC_RY',5X,
     6      'PLASTIC_T'/)
C
  980 FORMAT(I5,I11,1P,9E14.5)
  990 FORMAT(2(16X,1P,6E14.5/))
C
 1000 FORMAT(1X,'NODE',3X,'NODE ID.',
     111X,'EXX',11X,'EYY',11X,'EZZ',11X,'EXY',11X,'EYZ',11X,'EXZ'/)
C
 1010 FORMAT(1X,'NODE',3X,'NODE ID.',
     16X,' TOTAL_X    ',3X,' TOTAL_Y    ',3X,' TOTAL_Z    ',2X,
     2' TOTAL_XY    ',2X,' TOTAL_YZ    ',2X,' TOTAL_XZ    '/
     322X,' ELAST_X',6X,' ELAST_Y',6X,' ELAST_Z',5X,' ELAST_XY',5X,
     4' ELAST_YZ',5X,' ELAST_XZ'/
     522X,' PLAST_X',6X,' PLAST_Y',6X,' PLAST_Z',5X,' PLAST_XY',5X,
     6' PLAST_YZ',5X,' PLAST_XZ'/)
C
      END


C
C ========================== O U T 7 ==============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE OUT7(IOUT)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C DIRVEC
C
C*************************************************************
      REAL*4 SHELLZ
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1              NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC1(10),LDCONT,INCPTR
      COMMON/INPUTE/ISPB(10000)
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/MAIN2/UTOTAL(60000)
      COMMON/TRANS/DC(3 , 3)
      COMMON/SKTR2/SHELLZ(3 , 10000)
      COMMON/OUTPT2/IOEL(5000),IONOD(10000)
      DIMENSION DUMMY1( 6 )
C
```

```
C
      IF (IDIM .EQ. 3) THEN
          DO 110 K1 = 1, NNODES
              I = NNDF*(K1-1)
              ICODE = IAND(ISPB(K1),256)
              ISPS = IAND(ISPB(K1),4)
              I = I + IDIM
              IF (ICODE.GT.0 .AND. ISPS.NE.4) THEN
                  DC(1,3) = DBLE(SHELLZ(1,K1))
                  DC(2,3) = DBLE(SHELLZ(2,K1))
                  DC(3,3) = DBLE(SHELLZ(3,K1))
                  CALL DIRVEC (DC(1,1),DC(1,2),DC(1,3))
                  DO 100 K2 = 1, 3
                      CST = 0.
                      CST1 = 0.
                      IDIR = I + 1
                      CST1 = CST1 + UTOTAL(IDIR)*DC(K2,1)
                      IDIR = I + 2
                      CST1 = CST1 + UTOTAL(IDIR)*DC(K2,2)
                      IDIR = I + 3
                      CST1 = CST1 + UTOTAL(IDIR)*DC(K2,3)
                      DUMMY1(K2) = CST1
  100             CONTINUE
C
                  IDIR = I + 1
                  UTOTAL(IDIR) = DUMMY1(1)
                  IDIR = I + 2
                  UTOTAL(IDIR) = DUMMY1(2)
                  IDIR = I + 3
                  UTOTAL(IDIR) = DUMMY1(3)
              ENDIF
  110     CONTINUE
      ENDIF
C
      WRITE (IOUT, 910)
      DO 120 K1 = 1, NNODES
          IF (BTEST(ISPB(K1),10)) THEN
              ID = NNDF*(K1-1)
              IF (IONOD(K1) .EQ. 1) WRITE (IOUT, 900) K1, (UTOTAL(ID+K2)
     1            , K2 = 1, NNDF)
          ENDIF
  120 CONTINUE
C
      RETURN
  900 FORMAT(I11,1P,6E13.5)
  910 FORMAT(/1H1,20X,'N O D A L   T R A N S L A T I O N S   A N D  ',
     1        'R O T A T I O N S'/1X,' NODE NO.',
     2        11X,'UX',11X,'UY',11X,'UZ',11X,'RX',11X,'RY',11X,'RZ'/)
      END
C


C ========================== O U T 8 ===============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE OUT8(IOUT)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C DIRVEC
```

```
C
C************************************************************
      REAL*4 SHELLZ
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1              NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC1(10),LDCONT,INCPTR
      COMMON/INPUTE/ISPB(10000)
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/MAIN4/RE(60000)
      COMMON/TRANS/DC(3 , 3)
      COMMON/SKTR2/SHELLZ(3 , 10000)
      COMMON/OUTPT2/IOEL(5000),IONOD(10000)
      DIMENSION DUMMY( 6 )
C
C
      IF (IDIM .EQ. 3) THEN
          DO 110 K1 = 1, NNODES
              I = NNDF*(K1-1)
              ICODE = IAND(ISPB(K1),256)
              ISPS = IAND(ISPB(K1),4)
              I = I + IDIM
              IF (ICODE.GT.0 .AND. ISPS.NE.4) THEN
                  DC(1,3) = DBLE(SHELLZ(1,K1))
                  DC(2,3) = DBLE(SHELLZ(2,K1))
                  DC(3,3) = DBLE(SHELLZ(3,K1))
                  CALL DIRVEC (DC(1,1),DC(1,2),DC(1,3))
                  DO 100 K2 = 1, 3
                      CST = 0.
                      CST1 = 0.
                      IDIR = I + 1
                      CST = CST + RE(IDIR)*DC(K2,1)
                      IDIR = I + 2
                      CST = CST + RE(IDIR)*DC(K2,2)
                      IDIR = I + 3
                      CST = CST + RE(IDIR)*DC(K2,3)
                      DUMMY(K2) = CST
  100             CONTINUE
C
                  IDIR = I + 1
                  RE(IDIR) = DUMMY(1)
                  IDIR = I + 2
                  RE(IDIR) = DUMMY(2)
                  IDIR = I + 3
                  RE(IDIR) = DUMMY(3)
              ENDIF
  110     CONTINUE

      ENDIF
C
      WRITE (IOUT, 900)
      DO 120 K1 = 1, NNODES
          IF (BTEST(ISPB(K1),10)) THEN
              ID = NNDF*(K1-1)
              IF(IONOD(K1).EQ.1)WRITE(IOUT,910)K1,(RE(ID+K2),K2=1,NNDF)
          ENDIF
  120 CONTINUE
C
      RETURN
  900 FORMAT(/1H1,20X,'N O D A L   E Q U I L I B R I U M   F O R C E S',
     1       '  A N D   M O M E N T S'/'  NODE NO.',
     2        11X,'FX',11X,'FY',11X,'FZ',11X,'MX',11X,'MY',11X,'MZ'/)
  910 FORMAT(I11,1P,6E13.5)
      END
```

```
C
C ========================= O U T 9 ===============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE OUT9(IOUT)
C
C ================================================================
C I
C I   SUBROUTINE GETSTR ASSEMBELS THE GLOBAL STIFFNESS MATRIX AND/OR
C I   STORES THE NODE NUMBERS OF THE CORENT ELEMENT AND THE POSITION
C I   OF THE ELEMENT MATRICES IN THE GLOBAL MATRICES.
C I
C I
C I   II(J)    POSITION OF LOCAL STIFFNESS TERMS IN THE GLOBAL
C I            STIFFNESS MATRIX.
C I
C I   SKG(I) =    GLOBAL STIFFNESS MATRIX IN THE CONDENSED FORM
C I   SK(I,J)  =    ELEMENT STIFFNESS MATRIX
C I                 (SK IS COMPUTED BY SUBPROGRAM STIFEL)
C I
C I
C ===============================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C REWIN     ELINFO    ELINTM    LYINFO    ISH3DG    ISHSHL
C SHNORM    DIRVEC    IOGET     JACB3D    B3DLS     GETTHK
C JACSHL    BSHL      BTSHL     EQUILB
C
C*************************************************************
      INTEGER ELNUM
      CHARACTER*57 CTEMP
      REAL*4 SHELLZ,THICK
      REAL*8 LTHICK,N,NXI,NETA,NSI
      COMMON/LAYERB/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
      COMMON/UTIL1/STRESS(6),STRAIN(6),STRELA(6),CTEMP
      COMMON/ELSTR2/STRS(6)
      COMMON/SHLDIR/VECT(3,3,9)
      COMMON/INPUTE/ISPB(10000)
      COMMON/SKTR2/SHELLZ(3,10000)
      COMMON/ISHAP2/W(27)
      COMMON/MAIN4/RE(60000)
      COMMON/INPUT9/THICK(9),IFLAG
      COMMON/ISHAP1/N(20,27),NXI(20,27),NETA(20,27),NSI(20,27),SI(27)
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1              NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC1(10),LDCONT,INCPTR
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/INPUT2/NOP(20,5000)
      COMMON/ASSEM2/II(120)
      COMMON/FREEB/IFBODY(10000)
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/TRANS/DC(3 , 3)
      DIMENSION DUMMY( 6 )
      CALL REWIN

C ---- CONDENSED DATA STORAGE IN ARRAY IFBODY.
C
```

```
C ---- BIT STORAGE ORGANIZATION FOR IFBODY
C
C       BIT RANGE      ENTITY     DESCRIPTION
C
C       1-15           ELEMENT    0; DOES NOT BELONG TO FREEBODY
C                                 1; BELONGS TO FREEBODY
C       16-30          NODES      0; DOES NOT BELONGS TO FREEBODY
C                                 1; BELONDS TO FREEBODY
C
C       FOR ELEMENTS FREEBODY NUMBER IS THE BIT NUMBER.
C       FOR NODES FREEBODY NUMBER IS BIT NUMBER MINUS 15.
C
C ---- MDOF IS THE MAXIMUM NUMBER OF DEGREES OF FREEDOM
C
      MDOF = NNDF*NNODES
C


C ---- IFREE IS THE FREEBODY DIAGRAM NUMBER
C
      DO 220 IFREE = 1, NFREE
C
C ---- INITIALIZE THE EQUILIBRIUM LOAD VECTOR FOR EACH FREEBODY
C
          DO 100 K1 = 1, MDOF
              RE(K1) = 0.
  100     CONTINUE
C
C ----- NCB = NUMBER OF COLUMNS IN THE B MATRIX
C ----- NRB = NUMBER OF ROWS IN THE <B MATRIX
C ----- NNEL = NUMBER OF NODES IN THE ELEMENT
C
          DO 180 ELNUM = 1, NELEM
C
              CALL ELINFO (ELNUM, ITYPE, NNEL, IFLAG, ISTART, LINES)
              CALL ELINTM (ELNUM, IDENT, INTCOD, NIPXI, NIPETA, NIPSI,
     1            MATNUM, THICK)
C
C*VDIR: PREFER SCALAR
              DO 120 K1 = 1, NNEL
                  I1 = NNDF*(K1-1)
                  I2 = NNDF*(NOP(K1,ELNUM)-1)
C
C*VDIR: PREFER SCALAR
                  DO 110 K2 = 1, NNDF
                      K = I1 + K2
                      II(K) = I2 + K2
  110             CONTINUE
  120         CONTINUE
C
              DO 160 LYNUM = 1, NLAYRS
                  CALL LYINFO (LYNUM, LTHICK, ZS, MATRL, DCS, NNEL,
     1                ELNUM, NIPXI, NIPETA, NIPSI)
C
C ----- NENDF = NUMBER OF ELEMENT NODAL DEGREES OF FREEDOM
C ----- NCB   = NUMBER OF COLUMNS IN THE B MATRIX
C ----- NRB   = NUMBER OF ROWS IN THE B MATRIX
C ----- ITYPE = ELEMENT TYPE
C ----- IFLAG = ADDITIONAL IDENTIFIER FOR THE ELEMENT
C
                  IF (ITYPE .LE. 0) GO TO 170
                  IF (ITYPE.NE.0 .OR. IDENT.NE.0) THEN
                      IF (ITYPE .GT. 300) THEN
                          IF (IFLAG .EQ. 0) THEN
```

```
                  NCB = 3*NNEL
                  NRB = 6
                  NENDF = 3
                  CALL ISH3DG (ITYPE, NNEL, IERROR)
               ELSE IF (IFLAG .EQ. 4) THEN
                  NCB = 6*NNEL
                  NRB = 6
                  NENDF = 6
                  CALL ISHSHL (ITYPE, NNEL, IERROR)
               ENDIF
            ENDIF
         ENDIF
C     ITYPE1 = ITYPE
C     IDENT1 = IDENT
C
C ---- FOR SHELLS EVALUATE THE LOCAL SHELL COORDINATES OF THE NODES
C
         IF (BTEST(IFBODY(ELNUM),IFREE)) THEN
            IF (ITYPE.GT.300 .AND. IFLAG.EQ.4) THEN
               DO 130 K1 = 1, NNEL
                  KP = NOP(K1,ELNUM)
                  ICODE = IAND(ISPB(KP),4)
C
C ---- IF SHELL ROTATIONS ARE ASSEMBLED IN THE LOCAL SHELL COORDINATE
C       SYSTEM THEN RETRIEVE THE LOCAL Z-AXIS FROM STORAGE. ELSE
C       EVALUATE THE NORMAL TO THE SHELL MID PLANE BY A CALL TO THE
C       SHNORM ROUTINE.
C
                  IF (ICODE .GT. 0) THEN
                     CALL SHNORM (ELNUM, K1, NNEL, ITYPE,
     1                  VECT(1,1,K1),VECT(1,2,K1),VECT(1,3
     2                  ,K1))
                  ELSE
                     VECT(1,3,K1) = DBLE(SHELLZ(1,KP))
                     VECT(2,3,K1) = DBLE(SHELLZ(2,KP))
                     VECT(3,3,K1) = DBLE(SHELLZ(3,KP))
                     CALL DIRVEC (VECT(1,1,K1),VECT(1,2,K1)
     1                  ,VECT(1,3,K1))
                  ENDIF
  130          CONTINUE
            ENDIF
         ENDIF
C
C
C
         DO 150 INTGPN = 1, NIP
C
C ---- RETRIEVE STRESSES FROM STORAGE
C
            CALL IOGET (LDEV1, 96, '(A96)', 5)
C
C ---- CHECK TO SEE IF ELEMENT BELONGS TO FREEBODY DIAGRAM
C
            IF (BTEST(IFBODY(ELNUM),IFREE)) THEN
C
               DO 140 K1 = 1, 6
                  STRS(K1) = STRESS(K1)
  140          CONTINUE
C
C ----- GEOMETRICALLY LINEAR PROBLEMS
C
               IF (IFLAG1 .EQ. 0) THEN
                  IF (ITYPE .GT. 300) THEN
```

```
                              IF (IFLAG .EQ. 0) THEN
C
C ----- EQUILIBRIUM VECTOR FOR 3D LINEAR SOLID ELEMENTS
C
                              CALL JACB3D (INTGPN, ELNUM, NNEL,
     1                            IERROR, DETJAC)
                              CALL B3DLS (NNEL)
                              CST = DETJAC*W(INTGPN)
                              ELSE IF (IFLAG .EQ. 4) THEN
C
C ----- EQUILIBRIUM VECTOR FOR 3D LINEAR SHELL ELEMENTS
C
C ----- VECTOR V13 RETURNED BY JACSHL IS NORMAL TO MID SURFACE OF THE
C       SHELL AT INTEGRATION POINTS
C
                              CALL GETTHK (INTGPN, ELNUM, NNEL,
     1                            THICKE, RAD)
                              ISET = 0
                              SIP = SI(INTGPN)
                              CALL JACSHL (INTGPN, ELNUM, NNEL,
     1                            THICKE, DETJAC, V13, ISET, SIP)
C
C ----- THE COORDINATES VECTORS V11 AND V12 ARE EVALUATED BY DIRVEC
C       WHICH IS PART OF THE ELEMENT LIBRARY MODULE
C
                              CALL DIRVEC (V11, V12, V13)
                              CALL BSHL (NNEL, INTGPN)
                              CST = DETJAC*W(INTGPN)
                              CALL BTSHL (ELNUM, NNEL, 6, NRB, 3, 2)
                              ENDIF
                          ENDIF
                      ENDIF
                      CALL EQUILB (CST, NCB, NRB, NNDF, NENDF)
                  ENDIF
  150         CONTINUE
  160     CONTINUE
  170     CONTINUE
  180 CONTINUE
      IF (IDIM .EQ. 3) THEN
          DO 200 K1 = 1, NNODES
              IF (BTEST(IFBODY(K1),IFREE+15)) THEN
                  I = NNDF*(K1-1)
                  ICODE = IAND(ISPB(K1),256)
                  ISPS = IAND(ISPB(K1),4)
                  I = I + IDIM
                  IF (ICODE.GT.0 .AND. ISPS.NE.4) THEN
                      DC(1,3) = DBLE(SHELLZ(1,K1))
                      DC(2,3) = DBLE(SHELLZ(2,K1))
                      DC(3,3) = DBLE(SHELLZ(3,K1))
                      CALL DIRVEC (DC(1,1),DC(1,2),DC(1,3))
                      DO 190 K2 = 1, 3
                          CST = 0.
                          CST1 = 0.
                          IDIR = I + 1
                          CST = CST + RE(IDIR)*DC(K2,1)
                          IDIR = I + 2
                          CST = CST + RE(IDIR)*DC(K2,2)
                          IDIR = I + 3
                          CST = CST + RE(IDIR)*DC(K2,3)
                          DUMMY(K2) = CST
  190                 CONTINUE
C
                      IDIR = I + 1
```

```
                        RE(IDIR) = DUMMY(1)
                        IDIR = I + 2
                        RE(IDIR) = DUMMY(2)
                        IDIR = I + 3
                        RE(IDIR) = DUMMY(3)
                     ENDIF
                  ENDIF
  200          CONTINUE
            ENDIF
C
            WRITE (IOUT, 900) IFREE
            IF15 = IFREE + 15
            DO 210 K1 = 1, NNODES
               IF (BTEST(IFBODY(K1),IF15)) THEN
                  IF (BTEST(ISPB(K1),10)) THEN
                     ID = NNDF*(K1-1)
                     WRITE (IOUT, 910) K1, (RE(ID+K2), K2 = 1, NNDF)
                  ENDIF
               ENDIF
  210       CONTINUE
C
            CALL REWIN
  220 CONTINUE
C
      RETURN
  900 FORMAT(/1H1,20X,'N O D A L   E Q U I L I B R I U M   F O R C E S',
     1       '  A N D   M O M E N T S'/20X,'F O R   F R E E B O D Y',
     2       '  D I A G R A M ',I8/'  NODE NO.',
     3        11X,'FX',11X,'FY',11X,'FZ',11X,'MX',11X,'MY',11X,'MZ'/)
  910 FORMAT(I11,1P,6E13.5)
      END


C
C ========================= O U T 10 ============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE OUT10(IOUT)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C DIRVEC
C
C************************************************************
      REAL*4 SHELLZ
      INTEGER ELNUM
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1              NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC1(10),LDCONT,INCPTR
      COMMON/INPUTE/ISPB(10000)
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/MAIN4/RE(60000)
      COMMON/OUTPT2/IOEL(5000),IONOD(10000)
      COMMON/TRANS/DC(3 , 3)
      COMMON/SKTR2/SHELLZ(3 , 10000)
      DIMENSION DUMMY( 6 )
C
C
      IF (IDIM .EQ. 3) THEN
         DO 110 K1 = 1, NNODES
```

```
C
C ---- TEST BIT 18 OF ISPB TO SEE IF THE NODE IS A SUPPORT
C
              IF (BTEST(ISPB(K1),18)) THEN
                  I = NNDF*(K1-1)
                  ICODE = IAND(ISPB(K1),256)
                  ISPS = IAND(ISPB(K1),4)
                  I = I + IDIM
                  IF (ICODE.GT.0 .AND. ISPS.NE.4) THEN
                      DC(1,3) = DBLE(SHELLZ(1,K1))
                      DC(2,3) = DBLE(SHELLZ(2,K1))
                      DC(3,3) = DBLE(SHELLZ(3,K1))
                      CALL DIRVEC (DC(1,1),DC(1,2),DC(1,3))
                      DO 100 K2 = 1, 3
                          CST = 0.
                          CST1 = 0.
                          IDIR = I + 1
                          CST = CST + RE(IDIR)*DC(K2,1)
                          IDIR = I + 2
                          CST = CST + RE(IDIR)*DC(K2,2)
                          IDIR = I + 3
                          CST = CST + RE(IDIR)*DC(K2,3)
                          DUMMY(K2) = CST
  100                 CONTINUE
C
                      IDIR = I + 1
                      RE(IDIR) = DUMMY(1)
                      IDIR = I + 2
                      RE(IDIR) = DUMMY(2)
                      IDIR = I + 3
                      RE(IDIR) = DUMMY(3)
                  ENDIF
              ENDIF
  110     CONTINUE
      ENDIF
C
      WRITE (IOUT, 900)
      DO 120 K1 = 1, NNODES
          IF (BTEST(ISPB(K1),18)) THEN
              ID = NNDF*(K1-1)
              IF(IONOD(K1).EQ.1)WRITE(IOUT,910)K1,(RE(ID+K2),K2=1,NNDF)
          ENDIF
  120 CONTINUE
C
      RETURN
  900 FORMAT(/1H1,20X,'S U P P O R T   R E A C T I O N   F O R C E S',
     1        '   A N D   M O M E N T S'/'   NODE NO.',
     2         11X,'FX',11X,'FY',11X,'FZ',11X,'MX',11X,'MY',11X,'MZ'/)
  910 FORMAT(I11,1P,6E13.5)
      END
C
C ========================= O U T 11 ==============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE OUT11(IOUT)
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
```

```
C
C***********************************************************
      REAL*4 XYZ
      COMMON/INPUT3/XYZ(3,10000)
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1              NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC1(10),LDCONT,INCPTR
      COMMON/INPUTE/ISPB(10000)


      ID = 0.
C
      WRITE (IOUT, 900)
      DO 100 K1 = 1, NNODES
          IF (BTEST(ISPB(K1),10)) WRITE (IOUT, 910) K1, ID, XYZ(1,K1),
     1        XYZ(2,K1), XYZ(3,K1)
  100 CONTINUE
C
      RETURN
  900 FORMAT(/1H1,20X,'N O D A L   C O O R D I N A T E S'/
     1     2X,'NODE ID.',5X,'COORD NO.',12X,'X',12X,'Y',12X,'Z'/)
  910 FORMAT(I10,I14,1P,6E13.5)
      END
C
C ======================== E X T R A P ===========================
C
C     INCLUDE (PROCESS)
      SUBROUTINE EXTRP(VALUE)
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ELINFO    ELINTM    LAGRG1    IOGET     REWIN
C
C***********************************************************
      REAL*8 N
      REAL*4 THICK
      CHARACTER*1 YIELD
      INTEGER ELNUM
      COMMON/UTIL1/STRESS(6),STRAIN(6),STRELA(6),CENTER(6),WORK,YIELD
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/INPUT2/NOP(20,5000)
      COMMON/INPUTE/ISPB(10000)
      COMMON/INPUTF/MATYPE(10)
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/GRAPH2/IVS(12000),IVE(12000),ILS(12000),ILE(12000)
      COMMON/IREP1/IREP(12000),LREP(12000)
      COMMON/EXTRP1/INT33(9),INT22(4)
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1              NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC1(10),LDCONT,INCPTR
      COMMON/INPUT9/THICK(9),IFLAG
      COMMON/ISPC01/P(64 , 27)
      DIMENSION VALUE( * ),STRN(6,64),STRS(6,64),CAUC(6,64),
     1        CAUCH(6),AWORK(64)
C
C
      DO 100 K1 = 1, 14*NNODES
          VALUE(K1) = 0.
  100 CONTINUE
C
```

```
      ITYPE1 = 0
      IDENT1 = 0
      IEND = 0
      DO 250 ELNUM = 1, NELEM
          CALL ELINFO (ELNUM, ITYPE, NNEL, IFLAG, ISTART, LINES)
          CALL ELINTM (ELNUM, IDENT, INTCOD, NIPXI, NIPETA, NIPSI,
     1        MATNUM, THICK)
C
          IF (ITYPE .LE. 0) GO TO 240
          IF (ITYPE.NE.ITYPE1 .OR. IDENT.NE.IDENT1) THEN
              CALL LAGRG1 (ITYPE, IFLAG, NNEL, IERROR)
              IF (ITYPE .GT. 300) THEN
                  IEND = 6
              ELSE
                  IEND = 4
              ENDIF
          ENDIF
          ITYPE1 = ITYPE
          IDENT1 = IDENT
C
          DO 120 INTGPN = 1, NIP
              IF (MATYPE(MATNUM) .EQ. 1) THEN
                  CALL IOGET (LDEV1, 96, '(A96)', 5)
              ELSE
                  CALL IOGET (LDEV1, 201, '(A201)', 6)
                  AWORK(INTGPN) = WORK
              ENDIF
C
              DO 110 K1 = 1, IEND
                  STRS(K1,INTGPN) = STRESS(K1)
                  STRN(K1,INTGPN) = STRAIN(K1)
  110         CONTINUE
  120     CONTINUE
C
          DO 150 K1 = 1, NNEL


              NODE = NOP(K1,ELNUM)
              DO 140 K2 = 1, IEND
                  ID = (K2-1)*NNODES + NODE
                  DO 130 K3 = 1, NIP
                      VALUE(ID) = VALUE(ID) + STRS(K2,K3)*P(K3,K1)
  130             CONTINUE
  140         CONTINUE
  150     CONTINUE
C
          DO 180 K1 = 1, NNEL
              NODE = NOP(K1,ELNUM)
              DO 170 K2 = 1, IEND
                  ID = (IEND+K2-1)*NNODES + NODE
                  DO 160 K3 = 1, NIP
                      VALUE(ID) = VALUE(ID) + STRN(K2,K3)*P(K3,K1)
  160             CONTINUE
  170         CONTINUE
  180     CONTINUE
C
          DO 210 K1 = 1, NNEL
              NODE = NOP(K1,ELNUM)
              DO 200 K2 = 1, IEND
                  ID = (2*IEND+K2-1)*NNODES + NODE
                  DO 190 K3 = 1, NIP
                      VALUE(ID) = VALUE(ID) + CAUC(K2,K3)*P(K3,K1)
  190             CONTINUE
  200         CONTINUE
```

```
  210     CONTINUE
C
C     ID1 = 3*IEND*NNODES
C     DO 360 K1 = 1 , NNEL
C     ID = ID1 + NOP(K1 , ELNUM)
C     DO 360 K3 = 1 , NIP
C     VALUE( ID ) = VALUE( ID ) + VOLUMS( K3 )*P(K3,K1)
C360  CONTINUE
C
          ID1 = (3*IEND+1)*NNODES
          DO 230 K1 = 1, NNEL
              ID = ID1 + NOP(K1,ELNUM)
              DO 220 K3 = 1, NIP
                  VALUE(ID) = VALUE(ID) + AWORK(K3)*P(K3,K1)
  220         CONTINUE
  230     CONTINUE
  240     CONTINUE
  250 CONTINUE
C
      DO 270 K2 = 1, 14
          ID1 = (K2-1)*NNODES
          DO 260 NODE = 1, NNODES
              ITEST = IAND(ISPB(NODE),1024)
              IF (ITEST .GT. 0) THEN
                  IRNODE = IREP(NODE)/32
                  ID = ID1 + NODE
                  VALUE(ID) = VALUE(ID)/DFLOAT(IRNODE)
              ENDIF
  260     CONTINUE
  270 CONTINUE
C
      CALL REWIN
  280 CONTINUE
      RETURN
      END


C
C ===================== L A G R G 1 ================================
C
C     INCLUDE (PROCESS)
      SUBROUTINE LAGRG1(ITYPE,IFLAG,NNEL,IERROR)
C ===================================================================
C I
C I   P R O G R A M:
C I
C I   LAGRG1 EVALUATES THE LAGRANGE POLYNOMIAL COEFFICIENTS
C I   WHICH ARE USED FOR INTERPOLATING VALUES FROM GAUSSIAN
C I   INTEGRATION POINTS TO THE NODES. THIS SUBROUTINE MAY BE
C I   USED FOR QUADRILATERAL AND HEXAHIDRAL ISOPARAMETRIC ELEMENTS
C I   WHICH USE ANY ORDER OF GAUSS INTEGRATION.
C I
C I   A R G U M E N T   L I S T:
C I
C I   ITYPE   = INTERNAL ELEMENT TYPE NUMBER.
C I   IFLAG   = ANALYSIS TYPE FLAG.
C I   NNEL    = NUMBER OF NODES IN THE ELEMENT.
C I
C I
C I   O N    R E T U R N:
C I
C I   ARRAY P IN COMMON ISPC01 CONTAINS THE LAGRANGE INTERPOLATION
C I   POLYNOMIALS.
C I
```

```
C I
C ==========================================================
C
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C GAUSS
C
C***********************************************************
      REAL*4 AXI,AETA,XII,SII,ETAI
      REAL*8 P,XI,ETA,SI,PXI,PETA,PSI,WXI,WETA,WSI
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/ISPC01/P(64 , 27)
C
C ---- ELLIB1 CONTAINS THE COORD. OF THE NODES FOR HEXAHIDRAL
C        ELEMENTS IN THE NATURAL (ISOPARAMETRIC) COORDINATE SYSTEM.
C        THESE VALUES ARE SINGLE PRECISION REAL NUMBERS
C
      COMMON/ELLIB1/XII(20),ETAI(20),SII(20)
C
C ---- ELLIB2 CONTAINS THE COORD. OF THE NODES FOR QUADRILATERAL
C        ELEMENTS IN THE NATURAL (ISOPARAMETRIC) COORDINATE SYSTEM.
C        THESE VALUES ARE SINGLE PRECISION REAL NUMBERS
C
      COMMON/ELLIB2/AXI( 9 ),AETA( 9 )
      DIMENSION XI(4),ETA(4),SI(4),PXI(4),PETA(4),PSI(4)
      DIMENSION WXI(4),WETA(4),WSI(4)
C
C
C ---- NEXT IF BLOCK EVALUATES THE LAGRANGE POLYNOMIALS FOR THE
C        HEXAHIDRAL OR SHELL QUADRILATERAL ELEMENTS.
C
      IF (ITYPE .GT. 300) THEN
          CALL GAUSS (NIPXI, WXI, XI)
          CALL GAUSS (NIPETA, WETA, ETA)
          CALL GAUSS (NIPSI, WSI, SI)
          NIP = NIPXI*NIPETA*NIPSI
          I = NIPXI*NIPETA
C
C ---- EVALUATES THE LAGRANGE POLYNOMIALS FOR HEXAHIDRAL ELEMENTS.
C
          IF (IFLAG .EQ. 0) THEN
C
              DO 190 NODE = 1, NNEL
                  XIN = XII(NODE)
                  ETAN = ETAI(NODE)
                  SIN = SII(NODE)
C
                  DO 110 ISI = 1, NIPSI
                      PSI(ISI) = 1.0
                      DO 100 K1 = 1, NIPSI
                          IF (K1 .NE. ISI) PSI(ISI) = PSI(ISI)*(SIN-SI(
     1                        K1))/(SI(ISI)-SI(K1))
  100                 CONTINUE
  110             CONTINUE
C
                  DO 130 IETA = 1, NIPETA
                      PETA(IETA) = 1.0
                      DO 120 K1 = 1, NIPETA
                          IF (K1 .NE. IETA) PETA(IETA) = PETA(IETA)*(
```

```
     1                         ETAN-ETA(K1))/(ETA(IETA)-ETA(K1))
   120                 CONTINUE
   130             CONTINUE
C
                   DO 150 IXI = 1, NIPXI
                       PXI(IXI) = 1.0
                       DO 140 K1 = 1, NIPXI
                           IF (K1 .NE. IXI) PXI(IXI) = PXI(IXI)*(XIN-XI(
     1                         K1))/(XI(IXI)-XI(K1))
   140                 CONTINUE
   150             CONTINUE
C
                   DO 180 ISI = 1, NIPSI
                       DO 170 IETA = 1, NIPETA
                           DO 160 IXI = 1, NIPXI
                               INTGPN = (ISI-1)*I + (IETA-1)*NIPXI + IXI
                               P(INTGPN,NODE) = PSI(ISI)*PETA(IETA)*PXI(
     1                             IXI)
   160                     CONTINUE
   170                 CONTINUE
   180             CONTINUE
   190         CONTINUE
C
C
C ---- EVALUATE THE POLYNOMIALS FOR SHELL ELEMENTS. LOOP 110 ASSIGNS
C      A SI VALUE TO THE NATURAL COORDINTES OF THE NODES.
C      SI EQUALS TO -1., 0., AND 1. COORESPOND TO THE BOTTOM SURFACE,
C      MID-SURFACE, AND TOP SURFACE OF THE SHELL ELEMENTS.
C
         ELSE IF (IFLAG .EQ. 4) THEN
C
             SIN = -2.D0
             DO 300 ISURF = 1, 3
                 SIN = SIN + 1.0D0
                 DO 290 NODE = 1, NNEL
                     XIN = AXI(NODE)
                     ETAN = AETA(NODE)
                     DO 210 ISI = 1, NIPSI
                         PSI(ISI) = 1.0
C                WRITE(*,*)'ISI',ISI,'SI',SI(ISI)
                         DO 200 K1 = 1, NIPSI
                             IF (K1 .NE. ISI) PSI(ISI) = PSI(ISI)*(SIN-
     1                           SI(K1))/(SI(ISI)-SI(K1))
   200                   CONTINUE
   210               CONTINUE
C
                     DO 230 IETA = 1, NIPETA
                         PETA(IETA) = 1.0
                         DO 220 K1 = 1, NIPETA
                             IF (K1 .NE. IETA) PETA(IETA) = PETA(IETA)*
     1                           (ETAN-ETA(K1))/(ETA(IETA)-ETA(K1))
   220                   CONTINUE
   230               CONTINUE
C
                     DO 250 IXI = 1, NIPXI
                         PXI(IXI) = 1.0
                         DO 240 K1 = 1, NIPXI
                             IF (K1 .NE. IXI) PXI(IXI) = PXI(IXI)*(XIN-
     1                           XI(K1))/(XI(IXI)-XI(K1))
   240                   CONTINUE
   250               CONTINUE
C
                     DO 280 ISI = 1, NIPSI
```

```
                        DO 270 IETA = 1, NIPETA
                           DO 260 IXI = 1, NIPXI
                              INTGPN=(ISI-1)*I+(IETA-1)*NIPXI+IXI
                              P(INTGPN,NODE) = PSI(ISI)*PETA(IETA)*
     1                           PXI(IXI)
  260                      CONTINUE
  270                   CONTINUE
  280                CONTINUE
  290             CONTINUE
  300          CONTINUE
            ENDIF
         ENDIF
C
      RETURN
      END


C
C ======================== R E V I E W ==============================
C
C     INCLUDE (PROCESS)
      SUBROUTINE REVIEW(IDOF)
C
C ===================================================================
C I
C I   P R O G R A M:
C I
C I   I)
C I   REVIEW CHECKS THE VALIDITY OF NODE AND ELEMENT DEFINITIONS. IT
C I   ALSO GENERATES AUTO
C I   RESTRAINT INFORMATION FOR THE DRILLING DEGREES OF FREEDOM OF
C I   SHELL ELEMENTS. THE ROTATIONAL DEGREES OF FREEDOM OF THE NODES
C I   WHICH ARE SHARED BY SHELL ELEMENTS WITH THEIR MID-SURFACE
C I   NORMALS PARALLEL TO EACH OTHER ARE ASSEMBLED IN THE LOCAL
C I   COORDINATE SYSTEM OF THE NODE. THE
C I   Z-PRIME AXIS OF THE LOCAL COORDINATES IS NORMAL TO THE SURFACE
C I   OF THE SHELL. SINCE THERE IS NO STIFFNESS CONTRIBUTION TO THE
C I   ROTATIONAL DEGREE OF FREEDOM ABOUT THE Z-PRIME, THIS DEGREE OF
C I   FREEDOM  IS EXPLICITLY ELIMINATED BY SETTING THE APPROPRIATE
C I   TERM IN THE IDOF ARRAY EQUAL TO 2. FOR NODES THAT ARE
C I   RESTRAINED BY THE USER THE ROTATIONAL DEGREES OF FREEDOME IS
C I   ASSEMBLED IN THE GLOBAL COORDINATE SYSTEM. IN THIS SITUATION
C I   WHEN THE USER DEFINED RESTRAINED ROTATIONS HAVE NO COMPONENTS
C I   IN THE DIRECTION OF THE DRILLING DEGREE OF FREEDOM THE DRILLING
C I   DEGREE OF FREEDOM WILL BE RESTRAINED UNLESS THE NODE IS SHARED
C I   BY AT LEAST TWO OUT OF PLANE SHELL ELEMENTS. ALL NODES THAT ARE
C I   SHARED BY NON-TANGENT SHELL ELEMENTS WILL HAVE THEIR ROTATIONAL
C I   DEGREES OF FREEDOM ASSEMBLED IN THE GLOBAL REFERENCE FRAME.
C I   FOR THESE NODES SOME STIFFNESS EXISTS FOR ALL ROTATIONAL
C I   DEGREES OF FREEDOM, HENCE AUTO RESTRAINT IS NOT INVOKED.
C I
C I   II)
C I   IT DEFINES THE NODES THAT ARE RELATED TO FREEBODY DIAGRAMS.
C I
C I   O N   E N T R Y:
C I
C I   IDOF   = IS THE ARRAY CONTAINING THE NODAL RESTRAINT INFORMATION
C I            SPECIFIED BY THE USER.
C I             0; FREE NODE
C I             1; RESTRAINED WITH ZERO DISPLACEMENT OR ROTATION
C I            -1; RESTRAINED WITH NONZERO DISPLACEMENT OR ROTATION
C I
C I   O N   R E T U R N:
C I
```

```
C I   IDOF   = IS THE ARRAY CONTAINING THE NODAL RESTRAINT INFORMATION
C I            SPECIFIED BY THE USER AND BY THE SHELL AUTO RESTRAIN.
C I             0; FREE NODE
C I             1; RESTRAINED WITH ZERO DISPLACEMENT OR ROTATION
C I            -1; RESTRAINED WITH NONZERO DISPLACEMENT OR ROTATION
C I             2; RESTRAINED WITH ZERO ROTATION (AUTO RESTRAIN)
C I
C I
C ==================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ELINFO    ELINTM    ERRORS    SHNORM    DIRCOS    DOTPRO
C UNITVS
C
C************************************************************
      INTEGER ELNUM
      CHARACTER*6 COMM
      CHARACTER*7 CELEM,CNODE
      CHARACTER*80 BUFF,BUFFER
      REAL*4 XYZ,XAXIS,YAXIS,SHELLZ,THICK
      COMMON/INPUT2/NOP(20,5000)
      COMMON/INPUT3/XYZ(3,10000)
      COMMON/INPUT8/NNODES,NELEM,NEDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1            NINODE,NCOLOR,NFREE
      COMMON/INCR11/FRACT(10),NLINC1(10),LDCONT,INCPTR
      COMMON/INPUTD/XAXIS(4,10000),YAXIS(4,10000)
      COMMON/SKTR2/SHELLZ(3,10000)
      COMMON/TRANS/DC(3 , 3)


      COMMON/INPUTE/ISPB(10000)
      COMMON/COMP2/COMM,BUFFER,BUFF
      COMMON/FREEB/IFBODY(10000)
      COMMON/INPUT9/THICK(9),IFLAG
      DATA CELEM,CNODE/'ELEMENT','NODE   '/
C
C ----- ISPB( K ) STORES A SERIES OF CRITICAL INFORMATION ABOUT THE
C       NODES AND ELEMENTS. THE BIT STORAGE FOR EACH ELEMENT OF THIS
C       ARRAY IS AS FOLLOWS.
C
C       BIT 0    LOCAL COORDINATE SYSTEM FOR TRANSLATIONS
C       BIT 1    LOCAL COORDINATE SYSTEM FOR ROTATIONS IS DEFINED
C       BIT 2    0; SHELL ROTATIONS IN LOCAL SHELL COORDINATE SYSTEM
C                1; SHELL ROTATIONS ARE IN GLOBAL COORDINATE SYSTEM
C       BIT 3    SIGN BIT FOR THE THIRD DIRECTION COSINE OF THE
C                LOCAL TRANSLATION X-AXIS (0=+,1=-).
C       BIT 4    SIGN BIT FOR THE THIRD DIRECTION COSINE OF THE
C                LOCAL TRANSLATION Y-AXIS (0=+,1=-).
C       BIT 5    SIGN BIT FOR THE THIRD DIRECTION COSINE OF THE
C                LOCAL ROTATION X-AXIS (0=+,1=-).
C       BIT 6    SIGN BIT FOR THE THIRD DIRECTION COSINE OF THE
C                LOCAL ROTATION Y-AXIS (0=+,1=-).
C       BIT 7    SIGN BIT FOR THE THIRD DIRECTION COSINE OF THE
C                LOCAL SHELL ROTATION Z-AXIS (0=+,1=-).
C       BIT 8    0; NODE IS NOT SHARED WITH ANY SHELL ELEMENTS
C                1; NODE IS SHARED WITH AT LEAST ONE SHELL ELEMENT
C       BIT 9    0; NODE IS NOT SHARED WITH ANY ELEMENTS
C                1; NODE IS SHARED WITH AT LEAST ONE ELEMENT
```

```
C        BIT 10   0; POSITION OF THE NODE HAS NOT BEEN DEFINED
C                 1; POSITION OF THE NODE HAS BEEN DEFINED
C        BITS     1,0; CARTESIAN LOCAL TRANSLATION COORD. SYSTEM
C         11, 12  0,1; CYLINDRICAL LOCAL TRANSLATION COORD. SYSTEM
C                 1,1; SPHERICAL LOCAL TRANSLATION COORDINATE SYSTEM.
C        BITS     1,0; CARTESIAN LOCAL ROTATION COORD. SYSTEM
C         13, 14  0,1; CYLINDRICAL LOCAL ROTATION COORD. SYSTEM
C                 1,1; SPHERICAL LOCAL ROTATION COORDINATE SYSTEM.
C        BITS     1,0; CARTESIAN DEFINITION COORD. SYSTEM
C         15, 16  0,1; CYLINDRICAL DEFINITION COORD. SYSTEM
C                 1,1; SPHERICAL DEFINITION COORDINATE SYSTEM.
C        BIT 17   0; NODE IS NOT AN INTERFACE NODE.
C                 1; NODE IS AN INTERFACE NODE.
C        BIT 18   0; NODE IS NOT A SUPPORT
C                 1; NODE IS A SUPPORT
C        BIT 20   0; ELEMENT CONNECTIVITY HAS NOT BEEN DEFINED
C                 1; ELEMENT CONNECTIVITY HAS BEEN DEFINED
C
      DIMENSION IDOF(*),V1(3),V2(3),V3(3)
C
C ---- THE DEFAULT TOLORANCE ANGLE IS 3 DEGREES
C
C       TOLCOS  IS COSINE OF THE TOLORANCE ANGLE
C       ANGLE   IS THE TOLORANCE ANGLE IN RADIANS
C
      ANGLE = 3.0*0.017453292
      TOLCOS = DCOS(ANGLE)
      DO 170 ELNUM = 1, NELEM
          CALL ELINFO (ELNUM, ITYPE, NNEL, IFLAG, ISTART, LINES)
          CALL ELINTM(ELNUM,IDENT,INTCOD,NIPXI,NIPETA,NIPSI,MAT,THICK)
          IF (ITYPE .LE. 0) GO TO 160
C
C ---- TEST BIT 20 OF ISPB TO SEE IF THE ELEMENT CONNECTIVITY
C       HAS BEEN DEFINED.
C

          ITEST = IAND(ISPB(ELNUM),1048576)
          IF (ITEST .EQ. 0) THEN
              WRITE (BUFFER, *) CELEM, ELNUM
              CALL ERRORS (14, 0, 'REVIEW')
          ENDIF
C
          IF (ITYPE .GT. 300) THEN
              IF (IFLAG .EQ. 0) THEN
                  DO 100 K1 = 1, NNEL
                      NODE = NOP(K1,ELNUM)
C
C ---- TEST BIT 10 OF ISPB TO SEE IF THE NODE USED TO DEFINE
C       CONNECTIVITY HAS BEEN DEFINED.
C
                      ITEST = IAND(ISPB(NODE),1024)
                      IF (ITEST .EQ. 0) THEN
                          WRITE (BUFFER, *) CNODE, NODE, CELEM, ELNUM


                          CALL ERRORS (11, 0, 'REVIEW')
                      ENDIF
C
                      ISPB(NODE) = IBSET(ISPB(NODE),9)
  100             CONTINUE
              ELSE IF (IFLAG .EQ. 4) THEN
                  DO 140 K1 = 1, NNEL
C
C ---- TEST BIT 10 OF ISPB TO SEE IF THE NODE USED TO DEFINE
```

```
C       CONNECTIVITY HAS BEEN DEFINED.
C
                     NODE = NOP(K1,ELNUM)
                     ITEST = IAND(ISPB(NODE),1024)
                     IF (ITEST .EQ. 0) THEN
                         WRITE (BUFFER, *) CNODE, NODE, CELEM, ELNUM
                         CALL ERRORS (11, 0, 'REVIEW')
                     ENDIF
C
                     CALL SHNORM (ELNUM, K1, NNEL, ITYPE, V1, V2, V3)
C
C ---- SET BIT 8 OF THE ISPB ARRAY TO 1 (NODE IS SHARED BY A SHELL)
C
                     ISPB(NODE) = IBSET(ISPB(NODE),8)
                     ISPB(NODE) = IBSET(ISPB(NODE),9)
C
                     ID = NNDF*(NODE-1)
C
C ---- ISPS   =4; SHELL ROTATIONAL D.O.F. SHOULD BE IN GLOBAL COORD.
C             =0; SHELL ROTATIONAL D.O.F. SHOULD BE IN LOCAL COORD.
C ---- ICODE  =2; A LOCAL ROTATIONAL COORD. SYS. IS DEFINED FOR NODE
C             =0; ROTATIONAL COORDINATE SYSTEM AT THE NODE IS GLOBAL
C
                     ISPS = IAND(ISPB(NODE),4)
                     ICODE = IAND(ISPB(NODE),2)
C
C ---- THE FOLLOWING IF BLOCK IS USED TO DETERMINE IF ANY ROTATIONAL
C       RESTRAINTS ARE ASSOCIATED WITH THE NODE. IF RESTRAINTS ARE
C       PRESENT, THE SHELL ROTATIONS WILL HAVE TO BE ASSEMBLED IN THE
C       GLOBAL COORDINATE SYSTEM. IT SHOULD THEN BE CHECKED TO SEE IF
C       THE DRILLING DEGREE OF FREEDOM OF THE SHELL IS RESTRAINED
C       PROPERLY. THE ABOVE PROCESS IS NOT REQUIRED IF IT HAS PREVIOUSLY
C       BEEN DETERMINED THAT THE SHELL ROTATIONS SHOULD BE ASSEMBLED
C       IN THE GLOBAL REFERENCE FRAME.
C
                     IF (ISPS .EQ. 0) THEN
                         DO 110 K2 = 4, NNDF
                             ID1 = ID + K2
                             IF (IDOF(ID1).EQ.1 .OR. IDOF(ID1).EQ.(-1)
     1                           ) ISPS = 4
  110                    CONTINUE
C
                         IF (ISPS .EQ. 4) THEN
                             I = 0
                             DO 120 K2 = 4, NNDF
                                 I = I + 1
                                 ID1 = ID + K2
                                 IF (IDOF(ID1) .EQ. 0) THEN
                                 IF (ICODE .EQ. 0) THEN
                                 IF(DABS(V3(I)).GT.TOLCOS)IDOF(ID1)=2
                                 ELSE
                                 CALL DIRCOS (ICODE, IDIM, NODE)
                                 DOT = DOTPRO(V3,DC(1,I),3)
                                 IF(DABS(DOT).GT.TOLCOS)IDOF(ID1)=2
                                 ENDIF
                                 ENDIF
  120                        CONTINUE
                         ENDIF
                     ENDIF
C
C ---- IF MORE THAN ONE SHELL SHARE THE NODE, THEN THE SUM OF THE
C       NORMAL VECTORS IS ACCUMULATED IN SHELLZ ARRAY. THE DOT PRODUCT
C       OF THE CURRENT SHELL NORMAL V3 AND THE APPROPRIATE SHELLZ
```

```
C       COMPONENTS WILL DETERMINE IF THE CURRENT SHELL IS COPLANAR
C       (TANGENT) TO THE PREVIOUSLY PROCESSED SHELLS. IF SHELLS ARE NOT
C       COPLANAR THEN SUFFICIENT ROTATIONAL STIFFNESS EXISTS IN ALL
C       DIRECTIONS. HENCE, ALL THE IDOF COMPONENTS RESTRAINED BY THIS
C       MODULE (IDOF(N)=2) SHOULD BE SET EQUAL TO ZERO.
C
C
                    COSXZP = DBLE(SHELLZ(1,NODE))
                    COSYZP = DBLE(SHELLZ(2,NODE))
                    COSZZP = DBLE(SHELLZ(3,NODE))
C
                    DOT = V3(1)*COSXZP - COSYZP*V3(2) + V3(3)*COSZZP
                    CNORM = DSQRT(COSXZP**2+COSYZP**2+COSZZP**2)
                    IF (CNORM .NE. 0.) THEN
                       BCOS = DOT/CNORM
                    ELSE
                       BCOS = 1.
                    ENDIF
C
C
                    IF (ISPS .NE. 4) THEN
                       IF (DABS(BCOS) .GE. TOLCOS) THEN
                          IF (IDOF(ID+6) .EQ. 0) IDOF(ID+6) = 2
                       ELSE
                          ISPS = 4
                          IF (IDOF(ID+6) .EQ. 2) IDOF(ID+6) = 0
                       ENDIF
                    ELSE
                       IF (DABS(BCOS) .LT. TOLCOS) THEN
                          DO 130 K2 = 4, 6
                             ID1 = ID + K2
                             IF (IDOF(ID1) .EQ. 2) IDOF(ID1) = 0
  130                     CONTINUE
                       ENDIF
                    ENDIF
C
                    IF (ISPS .GT. 0) THEN
                       ITEMP = IBSET(ISPB(NODE),2)
                       ISPB(NODE) = ITEMP
                    ENDIF
                    SHELLZ(1,NODE) = SHELLZ(1,NODE) + V3(1)
                    SHELLZ(2,NODE) = SHELLZ(2,NODE) + V3(2)
                    SHELLZ(3,NODE) = SHELLZ(3,NODE) + V3(3)
C
C... NORMALIZE THE VECTOR SHELLZ
C
                    SNORM = SQRT(SHELLZ(1,NODE)**2+SHELLZ(2,NODE)**2+
     1                 SHELLZ(3,NODE)**2)
                    SHELLZ(1,NODE) = SHELLZ(1,NODE)/SNORM
                    SHELLZ(2,NODE) = SHELLZ(2,NODE)/SNORM
                    SHELLZ(3,NODE) = SHELLZ(3,NODE)/SNORM
  140            CONTINUE
              ENDIF
           ELSE IF (ITYPE .GT. 200) THEN
              DO 150 K1 = 1, NNEL
                 NODE = NOP(K1,ELNUM)
C
C ---- TEST BIT 10 OF ISPB TO SEE IF THE NODE USED TO DEFINE
C       CONNECTIVITY HAS BEEN DEFINED.
C
                 ITEST = IAND(ISPB(NODE),1024)
                 IF (ITEST .EQ. 0) THEN
                    WRITE (BUFFER, *) CNODE, NODE, CELEM, ELNUM
```

```
                    CALL ERRORS (11, 0, 'REVIEW')
                 ENDIF
C
                 ISPB(NODE) = IBSET(ISPB(NODE),9)
  150          CONTINUE
           ENDIF
  160      CONTINUE
  170 CONTINUE
C
C ---- IF A NODE IS NOT SHARED BY ANY SHELL ELEMENTS THEN RESTRAIN
C       THE ROTATIONAL DEGREES OF FREEDOM.
C
      DO 190 NODE = 1, NNODES
         IDENT1 = IAND(ISPB(NODE),512)
         IDENT2 = IAND(ISPB(NODE),256)
C
C ---- IF A NODE IS NOT SHARED BY ANY ELEMENT THEN RESTRAIN ALL DOF
C
         IF (IDENT1 .EQ. 0) THEN
             ID = NNDF*(NODE-1)
             DO 180 K1 = 1, NNDF
                 IDOF(ID+K1) = 1
  180        CONTINUE
C
C ---- IF A NODE IS NOT SHARED BY ANY SHELL ELEMENTS THEN RESTRAIN
C       THE ROTATIONAL DEGREES OF FREEDOM.
C
         ELSE IF (NNDF.EQ.6 .AND. IDENT2.EQ.0) THEN
             ID = NNDF*(NODE-1)
             IDOF(ID+4) = 1
             IDOF(ID+5) = 1
             IDOF(ID+6) = 1
             CALL UNITVS (SHELLZ(1,NODE),3)
         ENDIF
C
C ---- NORMALIZE THE SHELLZ VECTOR.
C
  190 CONTINUE
C
C*     DO 70 K1 = 1 , NNODES
C*     ID = NNDF*(K1 -1)
C
C
C ---- READ AND GENERATE FREEBODY INFORMATION FOR NODES
C
C ---- CONDENSED DATA STORAGE IN ARRAY IFBODY.
C
C ---- BIT STORAGE ORGANIZATION FOR IFBODY
C
C       BIT RANGE      ENTITY     DESCRIPTION
C
C       1-15           ELEMENT    0; DOES NOT BELONG TO FREEBODY
C                                 1; BELONGS TO FREEBODY
C       16-30          NODES      0; DOES NOT BELONGS TO FREEBODY
C                                 1; BELONDS TO FREEBODY
C
C       FOR ELEMENTS FREEBODY NUMBER IS THE BIT NUMBER.
C       FOR NODES FREEBODY NUMBER IS BIT NUMBER MINUS 15.
C
      DO 220 K1 = 1, NFREE
         IBITS = K1 + 15
         DO 210 ELNUM = 1, NELEM
             CALL ELINFO (ELNUM, ITYPE, NNEL, IFLAG, ISTART, LINES)
```

```
              IF (BTEST(IFBODY(ELNUM),K1) .AND. ITYPE.GT.0) THEN
                 DO 200 K3 = 1, NNEL
                    NODE = NOP(K3,ELNUM)
                    IFBODY(NODE) = IBSET(IFBODY(NODE),IBITS)
  200            CONTINUE
              ENDIF
  210     CONTINUE
  220 CONTINUE
C
      CALL ERRORS (0, 2, 'REVIEW')
      RETURN
      END


C
C ===================== U T I L I T ================================
C
C     INCLUDE (PROCESS)
      SUBROUTINE UTILIT
      IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C***********************************************************
      CHARACTER*201 BUFFER
      CHARACTER*6  FMAT
      COMMON/UTIL1/BUFFER
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/MAIN2/UTOTAL(60000)
      COMMON/MAIN4/RE(60000)
      COMMON/TEMP/PRESS,PLWORK
      COMMON/MPCS/COEFMP(40000),ALAMB(40000),MPCDOF(40000),
     $ MPCADR(2,5000),NMPC,MPCPNT,MAXMPC
      COMMON/ICONSA/IFLAG7
      DIMENSION IDOF( 1 )
c
C    THE ENTRYS IN THIS SUBROUTINE ARE USED TO SWAP, STORE AND RECOVER
C    INFORMATION USED IN THE NONLINEAR PROCEDURE.
C
C               E N T R Y     S W A P
C
      ENTRY SWAP
C
      ISWAP = LDEV1
      LDEV1 = LDEV2
      LDEV2 = ISWAP
C
      RETURN
C
C               E N T R Y     S W A P 1
C
      ENTRY SWAP1
C
      LDKEEP = LDEV1
      LDEV1 = LDEV2
      ISWAP = LDEV2
      LDEV2 = LDEV3
      LDEV3 = ISWAP
C
```

```
      RETURN
C
C              E N T R Y    S W A P 2
C
      ENTRY SWAP2
C
      LDEV1 = LDEV2
      LDEV2 = LDKEEP
C
      RETURN
C
C              E N T R Y    S W A P 3
C
      ENTRY SWAP3
C
      ISWAP = LDEV1
      LDEV1 = LDEV3
      LDEV3 = ISWAP
C
      RETURN
C
C              E N T R Y    R E W I N
C
      ENTRY REWIN
C
      REWIND (LDEV1, ERR=190, IOSTAT=IERROR)
      REWIND (LDEV2, ERR=190, IOSTAT=IERROR)
      REWIND (LDEV3, ERR=190, IOSTAT=IERROR)
      RETURN
C
C              E N T R Y    R E S T O R
C
      ENTRY RESTOR (MDF, ISTART, NTDF, IDOF)
C
      READ (LDEVST, *) ISTART, NTDF, NWMAX, LDEV1, LDEV2, LDEV3
      DO 100 K1 = 1, MDF
          READ (LDEVST, *) RE(K1), UTOTAL(K1)
  100 CONTINUE
      REWIND (LDEVST, ERR=190, IOSTAT=IERROR)
      RETURN
C
C              E N T R Y    S T O R E
C
      ENTRY STORE (FRACT, MDF, INCREM, NTDF, IDOF)
C
      WRITE (LDEVST, *) FRACT, INCREM, NTDF, NWMAX, LDEV1, LDEV2, LDEV3
      DO 110 K1 = 1, MDF
          WRITE (LDEVST, *) RE(K1), UTOTAL(K1), IDOF(K1)
  110 CONTINUE
      REWIND (LDEVST, ERR=190, IOSTAT=IERROR)
      WRITE (LDEV5, *) INCREM, PRESS, UTOTAL(2), UTOTAL(2114), PLWORK
      PLWORK = 0
      RETURN
C
C              E N T R Y    S T O R E 1
C
      ENTRY STORE1 (MDF, NTDF, IDOF)
C
      WRITE (15, *) NTDF, LDEV1, LDEV2, LDEV3


      DO 120 K1 = 1, MDF
          WRITE (15, *) RE(K1), UTOTAL(K1)
  120 CONTINUE
```

```
      REWIND (15, ERR=190, IOSTAT=IERROR)
      RETURN
C
C               E N T R Y    R E S T R 1
C
      ENTRY RESTR1 (MDF, NTDF, IDOF)
C
      READ (15, *) NTDF, LDEV1, LDEV2, LDEV3
      DO 130 K1 = 1, MDF
          READ (15, *) RE(K1), UTOTAL(K1)
  130 CONTINUE
      REWIND (15, ERR=190, IOSTAT=IERROR)
      RETURN
C
C               E N T R Y    I O G E T
C
      ENTRY IOGET (IDEV, LENGTH, FMAT, N)
      READ (IDEV, FMT=FMAT(1:N)) BUFFER(1:LENGTH)
      RETURN
C
C               E N T R Y    I O P U T
C
      ENTRY IOPUT (IDEV, LENGTH, FMAT, N)
      WRITE (IDEV, FMT=FMAT(1:N)) BUFFER(1:LENGTH)
      RETURN
C
C               E N T R Y    I O B K S
C
      ENTRY IOBKS (IDEV)
      BACKSPACE (UNIT=IDEV)
      RETURN
C
C               E N T R Y    A R C H I V
C
      ENTRY ARCHIV (MDF)
      RETURN
C
C               E N T R Y    R E C O V
C
      ENTRY RECOV (FRACT, MDF, ISTART, NTDF, IDOF)
C
      READ (LDEVST, *) FRACT, ISTART, NTDF, NWMAX, LDEV1, LDEV2, LDEV3
      DO 140 K1 = 1, MDF
          READ (LDEVST, *) RE(K1), UTOTAL(K1), IDOF(K1)
  140 CONTINUE
      REWIND (LDEVST, ERR=190, IOSTAT=IERROR)
      RETURN
C
C               E N T R Y    W C O N S T
C
      ENTRY WCONST
C
      DO 150 K1 = 1, 40000
          WRITE (4, *) COEFMP(K1), MPCDOF(K1)
  150 CONTINUE
      REWIND (4, ERR=190, IOSTAT=IERROR)
      DO 160 K1 = 1, 5000
          WRITE (15, *) EMPC, MPCPNT, MAXMPC, MPCADR(1,K1), MPCADR(2,K1)
  160 CONTINUE
      REWIND (15, ERR=190, IOSTAT=IERROR)
      RETURN
C
C               E N T R Y    R C O N S T
```

```
C
      ENTRY RCONST
C
      IF (IFLAG7 .EQ. 0) RETURN
C     READ(4 , *)NMPC,MPCPNT,MAXMPC
      DO 170 K1 = 1, 40000
          READ (4, *) COEFMP(K1), MPCDOF(K1)
  170 CONTINUE
      REWIND (4, ERR=190, IOSTAT=IERROR)
      DO 180 K1 = 1, 5000
          READ (15, *) NMPC, MPCPNT, MAXMPC, MPCADR(1,K1), MPCADR(2,K1)
  180 CONTINUE
      REWIND (15, ERR=190, IOSTAT=IERROR)
      RETURN
  190 CONTINUE
      WRITE (*, 900)
      STOP
  900 FORMAT(1H0,1X,'ERROR IN REWINDING UTILITY FILES IS DETECTED',
     1  ' BY ROUTINE CNTRL1')
      END


C
C====================== E R R O R S ============================
C
      SUBROUTINE ERRORS(NERR,NEXT,MODULE)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C***********************************************************
      INTEGER FMAT
      CHARACTER*6 COMM,MODULE,CODE
      CHARACTER*80 BUFFER,BUFF
      CHARACTER*70 TEXT
      COMMON/COMP2/COMM,BUFFER,BUFF
      DATA IERROR,IOUT/0,13/
      SAVE IERROR,IOUT
C
C     THIS SUBROUTINE IS SETUP TO REPORT ERRORS ENCOUNTERED DURING
C     INITIAL INPUT PROCESSING. NEED TO SET UP A FILE (UNIT 18) CONTAINING
C     ERROR MESSAGES.
C

      IF (NEXT .NE. 2) THEN
          IERROR = 1
          READ (18, REC=NERR) CODE, TEXT
          WRITE (IOUT, 900) CODE, MODULE, TEXT, BUFFER
C
          IF (NEXT .EQ. 0) THEN
              RETURN
          ELSE IF (NEXT .EQ. 1) THEN
              WRITE (IOUT, 910)
              STOP
          ENDIF
      ELSE IF (IERROR .EQ. 1) THEN
          WRITE (IOUT, 910)
          STOP
      ENDIF
C
```

```
  900 FORMAT(/1X,'------ FATEL ERROR: ',A6/
     1  9X,'Routine:',5X,A6/8X,A70/9x,'BUFFER CONTENTS ARE:'//A80)
  910 FORMAT(/1X,'------ PROGRAM TERMINATED DUE TO ONE OR MORE',
     1        ' FATAL ERRORS.')
      END
C     INCLUDE(PROCESS)
      SUBROUTINE IOCOPY
      IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C REWIN     SWAP
C
C*************************************************************
      CHARACTER*240 DUMMY
      CHARACTER*6   FMAT
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      CALL REWIN
C
      FMAT = '(A96)'
  100 CONTINUE
      READ (LDEV1, FMT=FMAT(1:5), END=110) DUMMY(1:96)
      WRITE (LDEV2, FMT=FMAT(1:5)) DUMMY(1:96)
      GO TO 100
C
C
  110 CONTINUE
      CALL SWAP
C
      RETURN
      END
```

Vita

Ananth Ramaswamy was born on 11th January 1963, in Bombay, India. He graduated from the Indian Institute of Technology Madras, India, in July 1985, with a Bachelor of Technology degree in Civil Engineering. In December of 1986 he completed his post graduate work at the University of California at Davis, earning a Master of Science degree in Engineering.

He joined the graduate program at Louisiana State University in August 1987. He is currently a candidate for the Ph.D degree in Civil Engineering at Louisiana State University, Baton Rouge.

DOCTORAL EXAMINATION AND DISSERTATION REPORT

Candidate: Ananth Ramaswamy

Major Field: Civil Engineering

Title of Dissertation: Nonlinear Inelastic Finite Element Analysis of Reinforced Concrete Structures with Emphasis on Shear and Torsion

Approved:

George Z. Voyiadjis
Major Professor and Chairman

Daniel Fogel
Dean of the Graduate School

EXAMINING COMMITTEE:

Vijaya K. A. Gopu

Patrick D. Zimanski

Date of Examination:

3/13/92